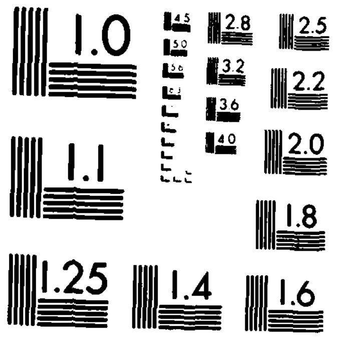

1.0
4.5
5.0
5.6
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

5

AD-A150 572

RADC-TR-84-189
Final Technical Report
September 1984

OCA MISSION PLANNING

Advanced Information & Decision Systems

Gregg Courand, Cindy O'Reilly and James R. Payne

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

RADC-TR-84-189 has been reviewed and is approved for publication.

APPROVED: Delores P.M. Clark

DELORES P.M. CLARK, 1LT, USAF
Project Engineer

APPROVED: [illegible]

RAYMOND P. URTZ
Technical Director
Command and Control Division

FOR THE COMMANDER: [illegible]

DONALD A. BRANTINGHAM
Plans Office

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A150572

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS N/A
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TR 3050-1	5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TR-84-189
6a. NAME OF PERFORMING ORGANIZATION Advanced Information & Decision System 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION Rome Air Development Center (COAD)
6c. ADDRESS (City, State and ZIP Code) 201 San Antonio Circle Suite 286 Mountain View CA 94040	7b. ADDRESS (City, State and ZIP Code) Griffiss AFB NY 13441-5700
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center 8b. OFFICE SYMBOL (If applicable) COAD	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F30602-83-C-0193
8c. ADDRESS (City, State and ZIP Code) Griffiss AFB NY 13441-5700	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: 65502F PROJECT NO.: 3005 TASK NO.: RA WORK UNIT NO: 10

11. TITLE (Include Security Classification)
OCA MISSION PLANNING

12. PERSONAL AUTHOR(S)
Gregg Courand, Cindy O'Reilly, James R. Payne

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM 10Mar84 TO 01Jun84	September 1984	140

16. SUPPLEMENTARY NOTATION
N/A

17. COSATI CODES

FIELD	GROUP	SUB. GR.
09	02	
15	07	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Hierarchical Systems
Hierarchical Planning
Offensive Counter Air Missions

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

In Phase I we have determined that existing decision analysis, artificial intelligence and operations research planning aids can be integrated in a research environment to provide a significant capability for addressing the overall hierarchical planning of tactical fighter air to ground missions. The aids selected during Phase I each address a different level of the offensive counter air (OCA) mission planning process: (1) Target Prioritization Aid (TPA) supports the user in determining the best enemy airfields to attack, given the number of friendly fighter bombers available; (2) KNOBS, a knowledge based system, provides an interactive capability for assigning aircraft and ordnance to each target in turn, and (3) Route Planning Air (RPA) provides for finding a nearly optimal route to the target, with respect to survival of the fighter. An architecture has been designed to incorporate these three kernel aids within an integrated planning tool. As part of this design effort, we have produced a theory of hierarchical software systems and derived a body of design principles.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ [✓ SAME AS RPT.] DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code)	22c. OFFICE SYMBOL
Delores P.M. Clark, 1Lt, USAF	315-330-7978	RADC (COAD)

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

CONTENTS

ILLUSTRATIONS

1. INTRODUCTION AND EXECUTIVE SUMMARY

1.1 MAJOR CONCLUSIONS

This project has investigated the feasibility of integrating several tools for planning Offensive Counter Air (OCA) missions. This work has proceeded along three tracks. a development of the underlying theory of hierarchical planning, an analysis of the requirements and behavior of existing individual aids (both as they stand alone and within an overall system), and the design of an architecture for combining an appropriate family of aids into an integrated tool. The theory of hierarchical planning provides a conceptual framework for integration, thus narrowing the range of possible architectures on the one hand and motivating many design decisions on the other. The data requirements and behavior of the aids further limit the range of possible designs. Our primary motivation has been to identify principles of hierarchical systems and of hierarchical system design. The design of a particular system is viewed both as an exemplification of these principles and as a useful problem-solving tool.

We have found that it is feasible to integrate the aids - the Target Prioritization Aid (TPA), the Knowledge-Based System (KNOBS), and the Route Planning Aid (RPA) - and thereby produce a valuable integrated OCA planning tool. In fact. there are a range of such integrated tools that can be designed, they differ according to the particular existing kernel aids chosen, as well as along a number of design dimensions - such as the degree of flexibility in accepting a given aids successor if and when it is developed.

There are a number of inherent problems that must be solved, regardless of which particular architecture is chosen. These problems/issues arise in two ways. First, there are hardware and software issues, machine, language, databases, and input/output formats. Second, there are the assumptions that the given aid makes about the world; what are the important objects, how to represent those objects, and how to use those objects (process them) to produce conclusions. It is apparent that while each aid can be expected to be internally consistent with respect to these two issues there is no reason to expect consistency among the individually developed aids. A great deal of attention was given to the architectural requirements for the resolution of the inconsistencies that arise when aids are joined. The basic requirements include aid-specific interfaces, which mediate the differences between any two aids, and an overall control regime which presents a unified view of OCA planning to the user.

For the remainder of this report, we will refer to the integrated tool as the Integrated Hierarchical OCA Planner, or simply OCAP.

1.2 HISTORICAL OVERVIEW OF PROJECT

We began our study of designing an integrated OCA planning tool with an analysis of the three aids - TPA KNOBS, and RPA. We reviewed the documentation sent to us and began to analyze alternative architectures for integration. In the first few months of the project, we also studied OCA planning in general (through visits to Blue Flag and the 9th Air Force at Shaw AFB). This provided us with a general framework in which to view each of the aids.

Our study of the aids was then expanded to include the Command and Control Warfare Strategy Planning Aid (CTA), the Dynamic Air Order-of-Battle Aggregation Aid (DAGR), and the Duplex Army Radio/Radar Targeting Aid (DART). We studied each of the aids in detail (through documentation only), characterizing their data base aspects, inputs and outputs, and user functions within the overall hierarchy of OCA planning. This study led us to a more complete understanding of the input-output relations of the various sets of aids that could be integrated.

After careful analysis of a set of criteria which we defined for candidate architectures, we narrowed our analysis of the aids to four candidate groups. (1) DAGR, DART CTA, TPA, (2) TPA, RPA; (3) TPA, KNOBS, RPA, and (4) DAGR, TPA, KNOBS, RPA. For each of these groups, we examined their data base and input-output issues, functions of the architecture, performance measures, intended user inter-model communication, interface issues, and evolution. After evaluating each of these against our set of criteria, we decided that the third group above - TPA, KNOBS, RPA - should be integrated into an integrated hierarchical OCA planner system (OCAP) for demonstration and evaluation.

We then began to design the OCAP in detail. In parallel, we developed an underlying theory of hierarchical planning. The theory provided us with a framework upon which to build a practical design; the design led us to ask (and answer) questions about the theory.

Finally our design led us to believe that the OCAP would indeed be a useful and valuable tool for OCA planners, and we have recommended that it be developed and re-evaluated.

1.3 RECOMENDATIONS

We recommend that the OCAP be built; TPA, KNOBS, and RPA should be integrated. It is also possible to include DAGR in the set of aids to be integrated; the advantages and disadvantages of this will be disussed in Section 5.3. This recommendation is based on three observations. First, an integrated OCA planning aid that concurrently accounts for all aspects (levels) of the OCA planning process should improve the quality of the resulting OCA plans. Second, the integration is feasible. Third, the technology of hierarchical planning promises to be a very powerful technique for problem solving. This is true even though the theory is young. We believe that building the OCAP would provide an opportunity to continue the elaboration of the theory of hierarchical system design that we have begun during this project.

We further recommend that the architecture be biased according to the stability of the aids over time. Thus, we explicitly recognize that KNOBS is likely at some point to be replaced by TEMPLAR, and that path planning tools tend to evolve quickly (to take advantage of increased computer power and to respond to improved capabilities for sensing enemy air defenses). The object of this bias is a commitment to stable integrated hierarchical aids and a concentration of effort in the most long-lived solutions to architectural problems. In this way, we expect OCAP itself to be stable and responsive to new individual aids.

As a final suggestion, the Air Force should begin to define the operational environment that would support and use OCAP, since the OCAP functions are not currently performed operationally in a well integrated manner

1.4 OVERVIEW OF REPORT

The report is organized as follows.

We first describe the OCA planning problem in Section 2. This description includes a discussion of the current environment and shows that there is a need for some kind of automated tool. We point out the advantages this kind of aid will have in the OCA planning environment We then introduce the OCAP system design issues.

Before discussing the OCAP in detail, we must first understand each of the six decision aids. We give a brief introduction to each of the aids in Section 3. This section describes the aids function, data bases, inputs, outputs, operation, and hardware/software.

Section 4 describes one of the major problems in integrating any set of individually developed software tools - data base consistency. We present a solution to the problem. This solution allows the aids to evolve and improve without too many changes to the OCAP.

In Section 5, we present the theory and preliminary design issues of the OCAP. We first discuss the theory of hierarchical systems. The theory gives us a framework for the design of an actual hierarchical planning tool. Next, we discuss the major design features of a hierarchical system. These design issues outline the features we must consider in designing an actual hierarchical planning tool, namely the OCAP. Keeping the theory just discussed in mind, we move to practical issues. We introduce candidate groups of aids which can be integrated into a planning tool. We evaluate each of them according to a set of criteria which we defined. Based on this evaluation we are led to believe that one candidate (namely, TPA, KNOBS, RPA) is best suited for integration.

Finally in Section 6 we give a detailed design of this candidate system. The design includes the definition of the OCAP's data bases, functions, user models, and interfaces.

2. PROBLEM DESCRIPTION OVERVIEW: OPERATIONAL AND TECHNICAL

There are two types of problems that we will address in this section. The first category of problems is domain dependent. Because of the extensive amount of bookkeeping, data gathering, and limited time, the current OCA planning environment cannot emphasize good planning as much as it should. An automated decision aid is needed. Once we decide we should build this tool, we are faced with the second type of problem: system design. How do we design a system which is both feasible from the computer science point of view and also valuable within the hierarchical planning environment? This section describes these problems.

2.1 OPERATIONAL OCA PLANNING PROBLEM

Current planning of Tactical Air Force missions, including Offensive Counter Air (OCA) missions, is done in a hierarchical organizational structure. The overall OCA objective each day is to produce a plan for the effective and efficient employment of tactical air resources available for OCA the next day. The planning is based on a large number of "small" decisions made at distributed locations. Each planner is expected to have done his job right and to have coordinated his efforts with others. Often the rationale for the small decisions is not subjected to adequate review and assessment, and there is little time to analyze and refine the one overall plan.

The OCA mission planning that is aided by the three aids we recommend integrating (TPA, KNOBS/TEMPLAR, RPA) is embedded in a much larger hierarchical planning environment that includes planning for all the types of tactical missions (OCA, Defensive Counter Air, Air Interdiction, Close Air Support). Figure 2-1 indicates nine levels in the planning hierarchy of functions that must be accomplished to generate an air tasking order. The functions within this hierarchy are performed by personnel in various organizations, not all of which are co-located. Each organization has its own data base and several of the functions in Figure 2-1 are performed to various degrees of completion by more than one organization.

The three aids have overlap in the planning hierarchy indicated in Figure 2-1. For example, both TPA and KNOBS include some of the targeteering, weaponeering and a number of sortie planning activities. The three aids taken together cover important aspects of many of the hierarchical activities such as target nomination, targeteering, weaponeering, aircraft assignment, and path selection.

The intent of hierarchically structured environments such as the military planning environment, is that guidance, constraints, and policy information be transmitted to lower levels and that solutions in line with that guidance be returned. Unfortunately, it is very easy and not uncommon for some of the guidance to be lost across the different levels of planning. There is no systematic process for checking the consistency of decisions with guidance provided nor for checking consistency of the parts of the plan with each other. In an effort to minimize the amount of guidance and to preserve the high-level priorities, the planning process has tended to be structured in such a way that flexibility is lost. This has improved plan coherence at the price of limiting the capability to develop, present, and compare a variety of employment options and their corresponding rationales.

1. Goal
   - Region
   - Enemy Function
2. Apportionment Nomination
   - Region
   - Mission Type
   - Over Time
3. Target Nomination
   - Priorities
   - Constraints
4. Targeteering
   - Target Component
   - Time of Attack
   - Terminal Air Defense Suppression
   - Target Attrition Level
   - Reattack Time
5. Weaponeering
   - Type
   - Number
6. Aircraft Types and Number of Sorties Nomination
   - Aircraft Penetration Model (Gross)
   - Packaging (EW, CAP SAM Suppression, Refueling)
7. Wing/Squadron Selection
   - Availability
   - Position
   - Time Over Target
8. Path Selection
   - Route Planning (Survival Optimization, Timing, TOT)
9. SAR

Figure 2-1: Planning Hierarchy Levels

Because distributed data bases are used and because the entire planning process takes most of the day, gathering accurate and consistent data is difficult, consumes much time and requires much "bean-counting" and "bookkeeping." Hence the form of the data does not support integrated planning either.

In summary, in the current operational environment bookkeeping is emphasized at the expense of planning.

2.2 SIGNIFICANCE OF PROVIDING AN INTEGRATED SET OF PLANNING AIDS

Major factors in improving tactical mission planning include accurate and common data, analysis and assessment across several levels of planning, and time. The development and use of an integrated set of automated planning aids for use at all levels of planning is needed and is a longer term goal. Initially, a man-machine system developed for the three primary organizational and functional levels, (i.e., target nominations aircraft/ordnance assignments, and flight path selections) should foster major improvements in the OCA mission plans. Its architecture should readily permit incorporation of aids for all levels of planning. These improvements include the maintenance of common, more accurate data bases, global analysis and exploration of alternatives, and explanation of rationales leading to the resulting OCA mission plans.

An appropriate integrated set of planning aids will provide analytical capability as well as free the decision makers from time-consuming "bookkeeping" details. This will allow them to concentrate more on trade-off issues and developing a "good" plan, rather than just getting "a" plan out by the deadline time. Thus, expected results include significant improvement in the effectiveness of OCA missions and improved survivability of the OCA air resources assigned. ("Effectiveness" refers not only to the plan for the individual OCA mission but also to the degree to which the overall OCA goals are being met. This last will be likened to measures of value derived from the overall plan and will lead to procedures that maximize that value.) Attaining the same increase in our capability to conduct successful OCA operations with the current OCA planning processes would probably require sizable increases in numbers of aircraft, personnel, and supplies.

2.3 SYSTEM-DESIGN PROBLEMS

Now that we have briefly described the OCA planning environment problem and indicated the advantages of designing an integrated set of planning aids, we discuss the problem of system design. We must design a man-machine integrated hierarchical planning system which is feasible ; whose value can be demonstrated in the current environment. For both these reasons it is prudent to choose a small set of aids that adequately span important hierarchical OCA planning functions. This section discusses these system-design problems.

2.3.1 Computer Science Issues

Several computer science issues must be addressed when we design the OCAP. The OCAP will include a subset of the six decision aids recently developed by RADC and shown in Figure 2-2. Probably the most important computer science issue arises from the fact that each of these six decision aids was developed more-or-less independently on a machine and language of the developer´s choice. Thus, the aids cannot simply be transferred to one machine and be expected to "understand" each other. Interfaces must be designed, inter-model data must be kept consistent, and an overall order of processing must be defined. All these tasks must be performed while keeping evolution of the OCAP and its component systems in mind. We do not want to design a system which will be out-of-date in a few years.

The above issues are addressed in subsequent sections of this report with possible solutions and procedures described.

2.3.2 Planning Issues

The OCAP we design must fit into the current OCA environment. Not only must it be a feasible design but it must be a useful one. Thus, the system must be hierarchical and address the functions currently performed by OCA planners. Thus, the interfaces we design to interact between the aids must be compatible with the aids' place in the hierarchy and must perform their tasks in the context of the current status of the planning process.

These issues constrain us to design a system which consists of only those aids which perform useful functions for the OCA planner and which together form a hierarchical planning system. (Other criteria for a useful integrated system are discussed in Section 5.3) The aids we recommend to integrate - TPA KNOBS, RPA - address these planning issues.

1.	TPA	-	A TARGET PRIORITIZATION AID. DETERMINES BEST SET OF TARGETS TO ATTACK.
2.	KNOBS	-	A KNOWLEDGE-BASED SYSTEM FOR ASSIGNING AIRCRAFT AND WEAPONS TO A NOMINATED TARGET.
3.	RPA	-	A ROUTE PLANNING AID. DETERMINES A MINIMUM LETHALITY ROUTE TO A TARGET.
4.	DAGR	-	AN AID FOR MAINTAINING THE CURRENT AIR ORDER OR BATTLE.
5.	DART	-	AN AID FOR MAINTAINING ENEMY AIR DEFENSE C^3 NODES.
5.	CTA	-	AN AID FOR PLANNING COUNTER-MEASURES AGAINST ENEMY C^3 CAPABILITIES THAT DEFEND AGAINST FRIENDLY OCA MISSIONS.

Figure 2-2: The Six Decision Aids

3. DESCRIPTION OF THE AIDS

This chapter contains descriptions of the six individual existing aids: TPA, KNOBS, RPA, CTA, DAGR, and DART. These aids comprise the set of individual existing "kernel" aids considered in designing the OCAP. Each aid is described in terms of its function, data bases, inputs, outputs, operation, hardware and software. (Parts of the descriptive material in each of the following subsections are taken from the Users Guide and Functional Description documents for the corresponding aid. See References.)

3.1 TARGET PRIORITIZATION AID (TPA)

3.1.1 Function of TPA

TPA assists the Target Nomination Officer within the Combat Operations Intelligence Division of a TACC. The aid allocates friendly sorties among a set of enemy air bases in order to reduce their sortie generation rate. TPA uses data base information and input from the user to calculate costs and benefits for the targets. TPA´s primary output is a list of recommended targets prioritized by their benefit-to-cost ratio.

TPA consists of three modules: (1) Target Nomination, (2) Expert Assessment, and (3) File Maintenance. The Target Nomination module conducts all of the planning; the Expert Assessment module allows the user to review and edit the expert's estimates used to calculate costs and benefits; and finally, the File Maintenance module allows the user to review and edit data bases.

3.1.2 Data Bases

TPA uses two main data bases: the air-base data base and the expert data bases. These data bases are described below.

3.1.2.1 Air-Base Data Base

The air-base data base consists of two data sets: the Air Installation File (AIF) and the Air Order of Battle (AOB). The AIF contains information for both Day 1 and Day 4 of the war.

The AIF contains the following data items for the current set of 50 Warsaw Pact air bases.

- o Air base ID
- o Location of the air base (longitude and latitude)
- o Number of observed targets for each component
- o Damage state of each component (there are five states)
- o Date of last update

Note that each air base is assumed to have ten components:

1. Aircraft in the Open
2. Maintenance Facilities
3. Aircraft in Shelters
4. Aircraft in Revetments
5. Command Posts
6. Control Sites
7. Launch and Recovery Surfaces
8. Petroleum, Oils and Lubricants
9. Munitions
10. Major Spares.

The AOB data set contains the names of 16 different Warsaw Pact aircraft and the type and number of each located at each air base.

3.1.2.2 Expert Assessment Data Base

The expert-assessed values and rationale used by TPA are stored in the Expert Data Base. This data is used to derive estimates of the costs and benefits associated with attacks on air base targets. The knowledge of experienced Air Force targeteers is reflected in these assessments.

TPA has space for six different expert data bases. Currently, only two data bases are available, but the user is free to create more. Thus, different experts can tailor the assessments to suit their own needs.

Each of the six data bases has six data types. TPA uses these data types to calculate costs and benefits with respect to a prototypical Warsaw Pact air base. This prototype represents a standard Warsaw Pact facility with a given number of units for each of the ten components.

The data types used by the prototype are the following: maximum effect, effects based on the AIF, effects of attacks over time, friendly sortie requirements, enemy aircraft average sortie generation rates, and enemy aircraft distance parameters.

Maximum effect is the percent reduction of an air base's sortie generation rate if the component is destroyed. This reduction is based on a 24-hour period. For example, if the maximum effect of a command post is 20, then if the command post is destroyed, the air base's sortie generation rate will be reduced by 20% for 24 hours.

Effects based on AIF are the percentages of maximum effect already achieved for each damage state. Thus, for a given damage state, this value represents how much of the maximum effect has been achieved.

The percentage of maximum effect achieved daily after a day's attack on a component is reflected in the effects of attack over time. This percentage is measured over 14 days, and thus degrades over that period.

Friendly sortie requirements are estimates of the number of F-111 equivalent sorties that could achieve maximum damage to a component with a 70 percent probability of success. There are estimates for each of the damage levels of each component.

Enemy sortie generation rates estimate the average number of sorties per day that each of the aircraft can maintain.

Finally, the enemy aircraft distance parameters represent the maximum flying distance the sortie generation rate can maintain and the maximum distance the sorties can fly.

3.1.3 Inputs

In addition to the data bases, TPA requires inputs from the user. The user enters two types of data. First, he can enter a record of the attacks flown in the past and those to be flown today. This history-of-attack data file provides the user with the opportunity to update the damage states of the air bases, if they have not yet been recorded in the AIF. Second, the user enters the scenario and conditions for the future. These conditions include the following: (1) weather forecast, (2) time window of effect, (3) enemy aircraft that must be suppressed, (4) region the user wants to protect, and (5) the number of friendly sorties available.

TPA allows the user to review and edit the data bases and target nomination information. Thus, by inputting responses to the system prompts, the user can edit virtually all of the data TPA uses. For a more detailed explanation of the many options available, see (Figgins, Gates, 1983) and (Waslov, 1982).

3.1.4 Outputs

TPA provides the user with system-wide outputs and Target Nomination outputs. Error messages and copies of screen displays are provided throughout each of the system's modes. The Target Nomination outputs include the tables of allocations of sorties, graphs of effects over time, and tables which trace various data. Examples of these outputs can be found in (Figgins, Gates, 1983) and (Waslov, 1982).

3.1.5 Operation

TPA leads the user through the system by a series of menus and prompts. After reviewing and editing the expert assessments and air base files, the user enters the Target Nomination mode. He selects which day of the war he is interested in - Day 1 or Day 4 - the expert assessment file he wants to use, and the type of planning in which he is interested. He can either edit the current conditions, conduct day-by-day planning, or conduct overall planning. Overall planning provides a quick multiple-day solution. Day-by-day planning allows the user to plan and analyze each day individually.

3.1.6 Hardware/Software

The following describes the hardware and software needed by TPA for the demonstration version currently running at PAR.

3.1.6.1 Hardware

The current version of TPA uses the following equipment:

o DEC VAX 11/780 under UNIX

o Any type of video or hardcopy terminal - better if capable of displaying more than 100 char/line, 42 lines/page. Currently TPA uses a Tektronix 4014 with an attached hardcopy device and a lineprinter. One terminal interface (DZ-11 or equivalent) is required.

- o Recommend use of a Floating Point Accelerator (DEC FP780/FP750) for the VAX.

3.1.6.2 Software

TPA requires the following software support:

- o UNIX APL/11 workspace in double precision format (Berkeley UNIX 4.1)

- o Uses the Purdue University modified version of UNIX APL, further enhanced by PAR.

3.2 KNOWLEDGE BASED SYSTEMS (KNOBS)

Parts of this section were taken directly out of KNOBS Architecture and Tactical Expert Mission Planner (TEMPLAR). See References.

3.2.1 Function of KNOBS

KNOBS was developed to support the design of individual strike missions within the air tasking process. Specifically, KNOBS supports Offensive Counter Air (OCA), mission planning. KNOBS has a representation of a typical OCA mission which the planner fills out with the details of a particular mission. Its primary goal is consistency management. It checks the constraints the user has inputted to insure they can be met, and informs the user of any inconsistencies.

KNOBS also provides planning support and autonomous planning facilities. It suggests alternatives for the resources, ranks the alternatives by their appropriateness, and is able to complete an entire mission design, given a few critical components.

3.2.2 Data Bases

The data base KNOBS uses currently contains information about 53 tactical units. These are distributed among eight air bases in Germany and Great Britain. The data base contains about 700 targets which are primarily located in East Germany. The speeds and ranges of 18 different types of aircraft are stored in the data base. It also contains information about the kinds of radar associated with several types of Soviet surface-to-air missiles.

The KNOBS data base also contains rules about the tactical doctrine for mission planning. KNOBS uses these rules to constrain or suggest possible choices of mission parameters.

The KNOBS data base represents a plausible scenario, but is not meant to be accurate. It is not classified.

3.2.3 Inputs

The minimum input to KNOBS is the target the planner wants to hit, the time-over-target, and the value of probability of kill expected (Pk). If only these inputs are entered, then KNOBS will search its data bases for candidates for all the other mission parameters, and fill in the remaining details, if necessary.

The user can also enter the other mission components: defenses, type of aircraft to be sent, location of those aircraft, ordnance, possible need for air-to-air refueling, and various call signs and frequencies for communication.

The user is also able to provide inputs in the form of data base queries. This capability is augmented by the uses of the natural language interface. This interface can construct small search procedures based on the user´s English input.

3.2.4 Outputs

KNOBS provides output to the user by providing planning support and autonomous planning facilities. It suggests alternatives, checks constraints, and completes entire mission designs. The KNOBS system explains its behavior to the user by monitoring its deduction mechanisms and translating a trace of their actions into English. It continually informs the user when constraints are not met and can conduct a limited dialogue with the user.

3.2.5 Operation

KNOBS contains a hierarchy of command environments. At the top of the hierarchy, KNOBS presents the user with a menu. The user can begin to plan or replan a mission, delete a mission, or quit. If he chooses to plan or replan a mission, the system will put him in the mission instantiation level.

The user has a few options at the mission instantiation level. He can select a mission item to fill or enter any of four different command modes.

If he selects an item, the system will prompt for a value. The user can restrict its allowable values rather than constraining it to a particular one. KNOBS uses these restrictions to constrain automatic planning. At any time, the user can enter one of the command modes to help him fill in items. He can request automatic planning by asking the system to "FINISH" the plan. KNOBS then prints a trace of its search for likely candidates.

The command modes the user can enter are control, explanation, enumeration, and English. If he wants to exit from the current mission, he should enter the control mode. At the explanation mode, he can ask the system to explain the available options or to explain problems with an item. If he wants to see a list of the acceptable choices for an item, the user can enter the enumeration mode. In the English mode, he can ask questions and enter the rule-editing facility.

3.2.6 Hardware/Software

KNOBS runs on a TOPS-20 system using INTERLISP.

3.3 ROUTE PLANNING AID (RPA)

3.3.1 Function of RPA

The Route Planning Aid (RPA) is designed to assist the F-111 mission planner to determine a minimum lethality route from friendly airspace to a specific target. RPA provides the following capabilities:

- o Derivation of minimum lethality routes - routes with maximum probability of survival,
- o User input of routes,
- o Route evaluation, and
- o Interactive iterative planning - the user can repeatedly modify a route during any given run.

The RPA system consists of three processing components: (1) the executive, (2) the optimization/evaluation system, and (3) the knowledge-based system. The executive manages file transfers between the other two components and provides a friendly user interface. The optimization/evaluation component determines minimum lethality routes and evaluates user-chosen routes with respect to lethality. Finally, the knowledge-based system provides the user with explanations and critiques about various parts of the route.

3.3.2 Data Bases

RPA uses four major data bases: (1) map, (2) threat, (3) target, and (4) aircraft. These data bases are described below.

3.3.2.1 Map Data Base

The map data base is in the form of DMA data of altitude contours. RPA uses this terrain information to calculate terrain masking effects. The terrain data is stored at five fixed altitudes above sea level.

3.3.2.2 Threat Data Base

The threat data base represents the information about the location and types of threats. Each threat is specified by type and location. Both fixed and mobile threats can be included.

3.3.2.3 Target Data Base

The targets in the scenario are specified in the target data base by longitude and latitude. The user must enter one such target when he runs RPA.

3.3.2.4 Aircraft Data Base

The F-111's minimum turn radius allowable and the maximum climb and dive rates are stored in the aircraft data base. Note that if any other type of aircraft is considered, these values will have to be changed.

3.3.3 Inputs

In addition to the four data bases, RPA needs additional inputs from the user. These inputs consist of the mission constraints and information about the route.

3.3.3.1 Mission Constraints

Mission constraints consist of region of interest, time-on-target, and several mission conditions. The region of interest defines the boundaries of the map displays that will appear on the graphics terminal. These boundaries are in the form of longitude and latitude coordinates. The user then enters time-on-target, followed by a list of mission conditions.

The mission conditions RPA uses are the following: time of day, visibility, jamming support, speed, and clearance altitude. The first three of these affect the threateningness of some of the threats. The lethality of the threat is increased if the mission is at night, visibility is poor, or there is no jamming support. Speed and clearance altitude must be entered and are held constant throughout the RPA run.

3.3.3.2 Route Information

In addition to the mission conditions which the user enters at time of initialization, he also inputs route information during the course of an RPA run. Of course, he must choose the target. He can also input his own route, if he is not satisfied with the minimum lethality route.

3.3.4 Outputs

The outputs of the RPA system are the following:

- o Intermediate output which another component uses as input,
- o Explanation and critiques,
- o Displays, and
- o A log of user interaction.

The optimization/evaluation component of RPA produces information about routes such as probability of survival, distance, altitude, etc. This intermediate output is passed on to the knowledge-based system to form explanations and critiques. These explanations provide details to the user about various parts of the route. The textual critiques accompany the displays which are shown on the color graphics terminal. RPA provides displays for terrain, lethality, and threats. Finally, all of the interaction between the user and the system is stored in a file which the user can access.

3.3.5 Operation

The operation of RPA can be described as a sequence of steps the user performs interactively.

First, the user initializes the system by entering all of the mission constraints.

Second, the user can receive a threat briefing. He can pose questions to the system about the particular types of threats in the scenario. He is also able to query the system about threats later during the run.

Next, the user focuses his attention on the target area. He can either choose an initial point (IP) or let the system choose one, based on minimum lethality.

Then, the user chooses an exit point and re-entry point on the friendly side of the FEBA. The system then determines the minimum lethality route based on these points. The user is free to enter his own routes at this point.

Finally, the user chooses an egress route.

3.3.6 Hardware/Software

The following description of the hardware and software needed by RPA pertains only to the demonstration version currently running at PAR Technology.

3.3.6.1 Hardware

The current version of RPA uses the following equipment:

o DEC VAX 11/780 under VMS

o Two displays devices

- a color graphics display device (currently a Ramtek)
- an alphanumeric device - any device that can display display ASCII characters and as many lines of text as possible.

3.3.6.2 Software

The RPA software requires the following software support:

o FORTRAN 77 compiler under VMS

o INTERLISP (current version was developed by SRI and ISI) Any changes in the current compiler would cause certain LISP functions to operate incorrectly.

3.4 COMMAND AND CONTROL WARFARE STRATEGY PLANNING AID (CTA)

3.4.1 Function of CTA

CTA was designed to assist the TACC personnel in applying C3CM strategy objectives to practical situations. The aid allows the user to perform the following functions:

- systematically consider various goals
- apply conditional restraints
- compute cost/benefit values for activities and make comparisons
- choose best options
- develop a C3 plan
- coordinate the plan implementation

Currently, CTA only considers the denial of command and disruption of control. Thus, it considers the use of weapons for destruction, jamming for isolation, and tactical deception.

3.4.2 Data Bases

CTA's data base (or "knowledge base") contains information pertaining to the mission of supporting an air base attack.

The following information is stored:

- A formal definition of the C3CM MISSION: in support of an air base attack.

- Division of the mission into 3 planning OBJECTIVES:

    1. Deny command
    2. Decrease control
    3. Condition command perception.

- For each objective, the desired OUTCOMES which could contribute to meeting it.

  There are 8 desired outcomes:
    1. Remove Command
    2. Isolate Command
    3. Overload Control
    4. Divert Sensors
    5. Navigational Aid Interference
    6. False Info to Air Defense Battle Staffs
    7. False Info to Parent Command
    8. Deceive into Early Interceptor Release

- The names and definitions of the 47 different kinds of ACTIONS available to cause the desired outcomes.

- For each action, the general KINDS OF ATTACK, the cost of the actions based on resources, and the outcomes influenced by the action.

  The five kinds of attack are:
    1. Physical Destruction
    2. Communications Jamming
    3. Radar Jamming
    4. Electronic Deception
    5. Physical Deception

- Potential contribution of the objectives, their outcomes and actions.

- Definitions of any conditions that might affect the assessments. The reasons for the effects.

- For each action, the recommended coordination activities.

3.4.3 Inputs

User input includes the following types of information:

- the kinds of attack that are available
- current status of conditions
- selection or rejection of specific actions
- whether he's working on a new or old plan
- whether he wants to save the plan

3.4.4 Outputs

The CTA system features a number of displays available to the user. The basic displays are as follows:

- a set of actions divided into three categories - Selected, Rejected, Undecided. This display also shows the total cost and benefit levels for the Selected actions.

- the current status of several intermediate goals and the degree to which the could be achieved if all Undecided actions were Selected.

- the impact of the current plan if a given action is selected or not.

- prioritized actions based on the difference in value caused by including or excluding each action from the plan.

- names of conditions and possible settings. This display highlights the current value of the conditions.

A number of textual outputs of the system are also available. These include explanations of action, conditions, goals, effects, impacts, and conditions.

3.4.5 Operation

CTA leads the user through the system through a series of menus and function keys. A main control menu is presented to the series at the start of any given run. This menu presents ten options. The major modes of operation are as follows:

1. Review/Change Plan.

   In this mode, the user can move actions from one category (Rejected, Undecided, Selected) to another. He can ask the aid to prioritize actions, and ask for explanations.

2. Plan by Objectives.

   In the Plan-by-Objectives mode, the aid chooses objectives from each category which maximizes the impact on the overall benefit.

3. Plan by Priority.

   This mode is the most automatic. The user specifies the number of actions he wants to add to the plan. The aid selects a high-priority group of that size from the Undecided category and moves it to Selected.

3.4.6 Hardware/Software

3.4.6.1 Hardware

The developmental version of CTA runs on an IBM 4331 Group II. The CTA program occupies a 256K byte workspace. The aid was designed to run on an IBM 3279 terminal, which has twelve programmable function keys. CTA can run on other terminals, provided some routines are rewritten to substitute other keys for the programmable function keys. The graphical capabilities of the IBM 3279 are not used.

3.4.6.2 Software

A subsidiary operating system called CMS under the VM/FT operating system of the IBM 4331 is used for CTA. CTA is written in VS APL, version 3.1.

3.5 DYNAMIC AIR ORDER OF BATTLE (DAGR)

3.5.1 Function of DAGR

DAGR supports the Air Order of Battle (AOB) analyst in validating reports, recognizing redeployment of aircraft, and associating destroyed aircraft with bases of origin. Beginning with a baseline AOB data base, DAGR processes incoming intelligent reports, classifies them, checks for inconsistencies, and updates the data base to reflect the information. Currently, DAGR processes two types of reports - Kill Reports (KILLREPs) and Movement Summary Reports (MSRs).

The user is able to interact with DAGR at two places. First, the user is able to correct any errors in the intelligence reports. Second, DAGR recommends changes to the AOB but will not implement them without getting the user's approval. The user is able to examine any information in the system in order to make his judgment.

3.5.2 Data Bases

DAGR uses three data bases - two dynamic and one static.

One dynamic data base contains the hypothesis structure which is a collection of "units" containing data about an object. For example, there is a unit which records information about a reported kill. All information about the kill is stored in a record with a unique identifier generated by DAGR.

Information about the air bases being monitored is stored in the other dynamic data base. DAGR keeps track of 28 airfields. This file makes up the AOB. This information is continually updated on the basis of the KILLREPs and MSRs.

The static data base contains information on the aircraft. The following information is stored:

- Type
- Model
- Role
- Combat radius
- Low altitude speed
- Max speed
- Service ceiling
- Soviet aircraft designation
- Two letter abbreviation of aircraft used internally

DAGR uses this information to check the accuracy and consistency of incoming reports.

3.5.3 Input

Two types of input are required to the system. The user must correct faulty information or tell the system to disregard it. Also, the user must approve or disapprove DAGR's update recommendations to the AOB.

If the user wants to correct faulty information, he must enter the correction in exactly the same format DAGR uses internally. If the user makes any serious mistakes in his corrections, they will be noticed by DAGR, and the user will be asked to fix the information again.

When DAGR recommends an AOB change, it informs the user of the change and gives him four options. The user can (1) accept the suggestion, (2) reject it, (3) ask to see evidence for the change, or (4) ask to see possible alternatives.

3.5.4 Output

When running DAGR, the user will be able to look at three different displays - (1) an alphanumeric display of incoming messages, (2) an alphanumeric display of AOB changes and other system output, and (3) a graphical display of the current situation. The incoming messages are simply the KILLREPs and MSRs. The second alphanumeric display contains DAGR's suggestions to the user and all prompts and explanations of changes. The graphical display is a map of the 28 airfields and tracks of groups of aircrafts and kills.

3.5.5 Operation

The operation of DAGR is simple. The incoming intelligence reports are displayed as described above. Unusable reports are brought to the user's attention and the user can correct it if he wishes. DAGR processes the messages, and when it determines a change should be made to the AOB, it informs the user. The user can respond as described in Section 3.5.3.

3.5.6 Hardware/Software

3.5.6.1 Hardware

DAGR runs on a DEC-20, using three terminals. It needs a hardcopy device (such as an Anderson-Jacobsen), an ASCII video-display terminal (such as a Datamedia 2500, DEC VT100), and a Tektronix 4014 with extended graphics.

3.5.6.2 Software

DAGR is written in INTERLISP-10 and requires the standard INTERLISP run-time environment.

3.6 DUPLEX ARMY RADIO/RADAR TARGETING AID (DART)

3.6.1 Function of DART

DART assists the C^3CM analyst by maintaining an up-to-the-minute C^3 Order of Battle File. The user can examine incoming messages from the Direct Support Unit (DSU), get advice on the message's meaning, and update the C^3OB data base. These incoming messages contain information about C^3 nodes - some communications signal, time of intercept, approximate location, and nature of the message's source. The C^3OB data base will be updated automatically by DART if the user approves of the system's advice. Otherwise, the user can update it himself.

3.6.2 Data Bases

Information on the current C^3 nodes is stored in the single data base maintained by DAGR - the C^3 Node Model. The information includes the following:

- a certainty factor indicating how sure the analyst is that the node actually exists

- the semi-major axis of the error ellipse associated with the location

- the names of any sensed ELINT messages associated with the location.

3.6.3 Input

All the user input to the system is in the form of responses to system prompts (except for initialization commands). The user can choose from seven options from the executive command menu:

1. Process next message
2. Get next message
3. Get advice about this message
4. Update model
5. Update display
6. Retrieve last message
7. Quit

These are described in Section 3.6.5.

3.6.4 Output

All system output is in response to user's queries. The system will display messages, explain advice about messages, output results of updating the model and update the color graphical displays.

3.6.5 Operation

After initializing the system, the user will be shown the Executive Command menu. This menu is the main contact between the user and the system. The options available to him were listed in Section 3.6.3.

The first option, "Process Next Message," is the most common. If chosen, this option prints the next message at the bottom of the screen and fills the rest of the screen with advice. The user can get more information by responding to the "Explain Advice" prompt. He is also able to examine other hypotheses.

The second option, "Get Next Message," simply retrieves the next message in the queue and displays it. To get advice about it, he must choose the third option "Get Advice About This Message."

The fourth option is to "Update the Model." The user can:

- automatically create a C^3 Node, or
  create a node himself.
- delete a node.
- automatically update a node, or
  update the node himself.
- find all nodes co-located with the node described in the current message.
- list all C^3 nodes of a particular type.
- quit.

The fifth option allows the user to update the graphical displays. Through this option, the user can display C^3 nodes and examine several useful pieces of information graphically.

The sixth option allows the user to "Retrieve Last Message" if he has skipped it. Finally, the last option is simply to quit and to stop DART execution.

3.6.6 Hardware/Software

3.6.6.1 Hardware

DART currently runs on a VAX 11/780 with a Floating Point Accelerator (FP780).

The terminal requirements consist of a Gould/DeAnza IP8500 display terminal (for output) with five memory boards. (DART needs three.) DART makes use of the alphanumeric overlay capability of the IP8500. Any CRT terminal with screen-oriented cursor control capabilities will suffice for input.

3.6.6.2 Software

DART was written in Pascal and C. Although it was developed under the UNIX operating system, it is not UNIX-dependent.

4. DATA BASE CONSISTENCY

When integrating independently developed software tools, one immediately faces the problem of data base consistency. Although there is some overlap between the data bases of some of the aids, the overlap is not nearly enough, especially within the particular subsets of aids most appropriate to integrate.

Each of the six OCA planning aids was developed on a machine and using a language of the developer's choice. Specifically, as noted in the previous section, TPA was developed on a VAX 11/780 (UNIX) using APL; KNOBS runs on a DEC-20 (TOPS-20) using INTERLISP; RPA runs on a VAX 11/780 (VMS) in FORTRAN; CTA uses APL on an IBM 4331; DAGR was developed on a DEC-20 in INTERLISP 10; and DART was written in Pascal and C on a VAX 11/780. These hardware and software differences cause problems when trying to ensure consistency.

Consider, for example, the subset including DAGR and TPA. Both need to access the Air Order of Battle (AOB) file, but each aid currently accesses its own version of it. DAGR's AOB data base is in the form of INTERLISP record declarations. TPA uses itemized lists of data for each air base and accesses these lists using the APL language. DAGR's main function is to update the AOB file. To be useful, TPA must access the most up-to-date version. Thus, these data bases must be kept separate but up-to-date. Thus, consistency must be maintained both across data base formats and over time. There must be some level of consistency management to reflect DAGR's updates to the AOB when TPA needs to access it.

It is important to note that the safest way to assure that each aid will use consistent information is to modify each of the aid's software so that each could access the same data base. But this option requires extensive rewriting of the existing programs. We consider this option undesirable at this time.

Several more practical options are available to ensure data base consistency. In fact, there are several levels of data base consistency that can be incorporated into the aids. Each level requires using a Consistency Manager. Consistency management in general is described next, followed by a description of the levels of integration possible.

4.1 CONSISTENCY MANAGEMENT

We define consistency management as a two-step process used to ensure data base consistency across all the aids' data bases.

Figure 4-1 and 4-2 illustrate this two-level process. In the first level (Figure 4-1), the human first defines the scenario context which is fed into the "Consistency Rule Builder" to build a set of consistency rules. The Consistency Rule Builder is a necessary tool because different scenario contexts imply different interpretations of particular concepts of consistency. Thus, a different set of rules is needed for different scenario settings. For example, the user needs to define such contexts as the location of the scenario (e.g., Middle East, Eastern Europe), type of war, level of war, etc. Weather is different when applied to the Middle East than when referring to Eastern Europe. A rule which maps "bad weather" used in one data base to "poor visibility" used in another would be different for these two locations. Type and level of war also require different rules depending on their particular contexts. Permissible weapons, for example, may vary from context to context.

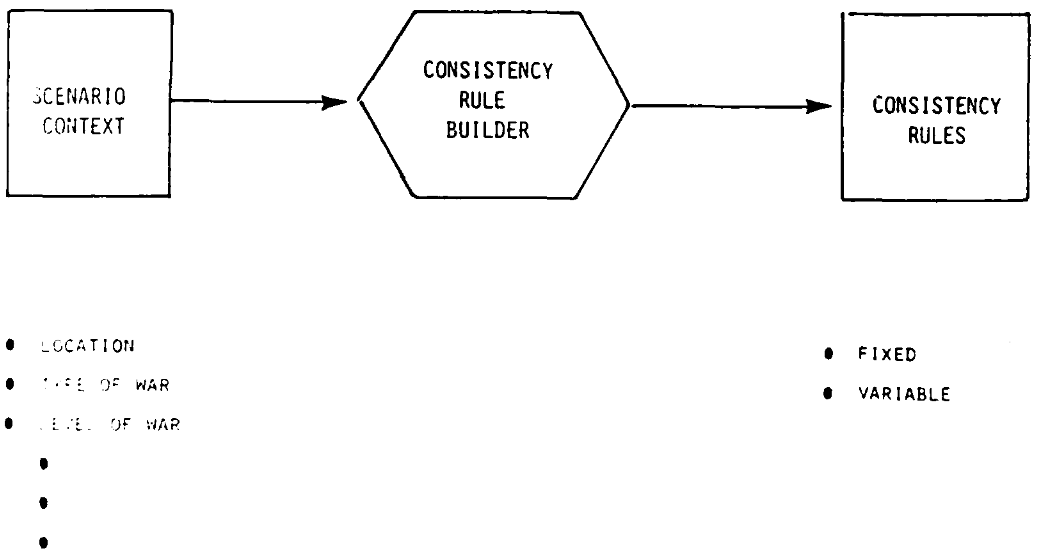


Figure 4-1: Consistency Management - Rule Building (1st Level)

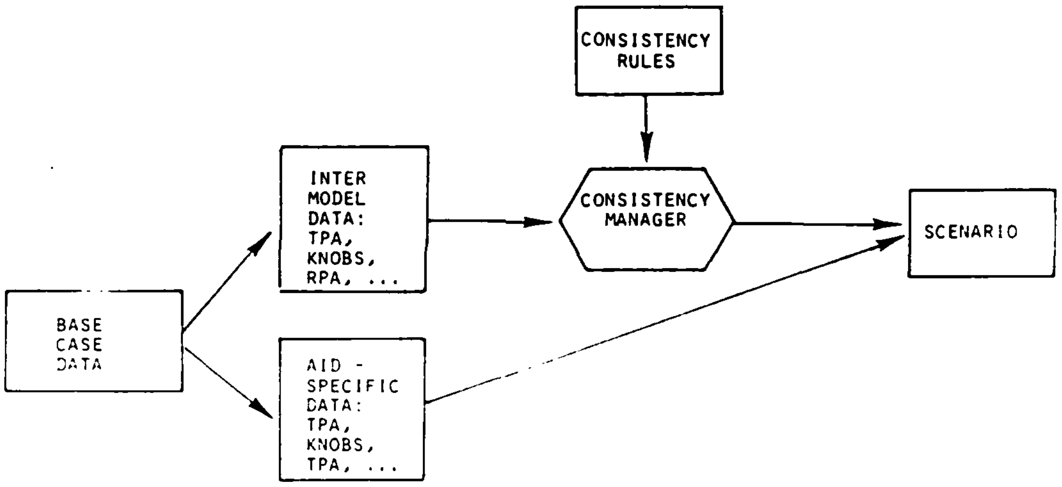

CONSISTENCY RULES
INTER MODEL DATA: TPA, KNOBS, RPA, ...
CONSISTENCY MANAGER
SCENARIO
BASE CASE DATA
AID - SPECIFIC DATA: TPA, KNOBS, TPA, ...

Figure 4-2: Consistency Management - Consistency Checking (2nd Level)

Some data pertaining to the scenario context will be fixed (such as location) throughout the human's use of the aid. Thus, he will enter this data only once, before using the integrated aid. This data will map into a set of fixed rules. Other types of data may vary from one set of runs to another (such as level of war). Thus, the user will have to enter this data as it changes to produce what we term "variable rules."

The Consistency Rule Builder would know the format of each data base and the representation of each data item to the various aids. The builder would develop a set of rules that map the representation of each data base item onto the representation of similar data in the other data bases. For example, one set of consistency rules would exist for weather. It might map the "W" used in TPA to represent bad weather onto the "bad_weather=true" value for KNOBS and the "P" for poor visibility used in RPA. Similar functions would exist for time of day, number of friendly sorties, etc. Of course, the data used by only one aid, such as the threat information RPA uses, would not need to be modeled by the Rule Builder.

The second level of the consistency process is depicted in Figure 4-2. This level defines what happens during each individual run. At the start of each run, the user supplies the base case data to the data bases of the integrated OCA tool. Base case data refers to the data particular to a given run. For example, the user would enter the particular targets of interest, sorties available, etc. Then, the data used by more than one aid (inter-model data) is passed from the specific data bases to the "Consistency Manager." The Consistency Manager, using the consistency rules developed above, would periodically check that data across all data bases is consistent. Probably, the Manager will take the most recent data as the "truth" and make all inconsistent data match that. Any inconsistencies which the Manager cannot resolve, would probably be brought to the user's attention. Consistent data is then collected to form the scenario data used by the integrated OCA planning system.

4.2 LEVELS OF CONSISTENCY MANAGEMENT

There are different levels of complexity at which we could design the Consistency Manager. The Manager could be human-aided or automatic. It could be intelligent enough to recognize when enough changes in one data base require re-running a specific aid. These levels are described next.

4.2.1 Human-Aided Consistency Manager

The first, most basic level of consistency management that can be designed is one which checks the consistency of data across data bases and simply informs the user of any differences it finds. Thus, in the DAGR-TPA example above, a simple Consistency Manager would determine that TPA's AOB file was different than DAGR's. The Manager would then inform the user. It would then be up to the user to resolve the differences (most likely by updating TPA's data base to match DAGR's).

This Consistency Manager is the most basic because it leaves all decisions up to the user and does not do any internal updating by itself. Although this Manager is the safest, due to the constant interaction with the human, it can also become cumbersome. Certainly, if the data item, such as "weather" had different values in two data bases, a human would be required to resolve it. But, in the DAGR-TPA example, because DAGR's main function is to update the AOB file, the Consistency Manager would frequently find mismatches between DAGR's data base and TPA's. It would be inefficient for the user to be required to update all of TPA's data base manually. Some degree of automated updating should be incorporated in the Consistency Manager.

4.2.2 Automated Consistency Manager

An automated consistency Manager would notice differences in data bases and resolve them according to some predetermined set of rules. One such rule would be, referring to the DAGR-TPA example again, to automatically update TPA's data base to reflect changes in DAGR's. It is clear, however, that many decisions would still require human interaction. Thus, some inconsistencies would still be brought to the user's attention to correct.

4.2.3 Intelligent Consistency Manager

Taking the level of complexity one step further, a very sophisticated Consistency Manager could be intelligent enough to recognize when specific aids should be run again and inform the user. If DAGR's data base changed significantly, this complex manager would not only realize that TPA's data base must be updated but that TPA actually should be run again.

This Manager is more complex than the others because it needs to recognize not only new information in a data base, but when that information is significant enough to merit re-planning. For example, the Manager might recognize when a "significant build-up" of enemy forces occurred to merit re-running TPA. The intelligent manager would still bring some data base inconsistencies to the user's attention if it could not resolve them itself.

4.3 RECOMMENDATIONS/SUMMARY

Maintaining separate data bases and building a Rule Builder and Consistency Manager requires no modifications to the existing software and allows the aids to access their own data bases in their current formats. New software must be developed, however, to build the Consistency Rule Builder and Manager. We believe this task to be a simpler and more manageable one than modifying the current aids. It also allows other aids to replace the current ones eventually regardless of the new aids' language or data base format. Evolution of the integrated aids is an important consideration. If an aid is replaced, the only change necessary to make the data bases consistent is the modification of the Consistency Rule Builder and Manager.

It should be noted at this point that consistency management is a long-term goal of the design of an integrated system. A first cut at maintaining consistency would be the human-aided Consistency Manager. This level still requires that consistency rules be built for the current scenario. To make this even simpler, possibly the human could define his own set of rules and manually enter them into the Consistency Manager. As the system advances, the Rule Builder can be developed to define the set of rules automatically.

For the remainder of this report, we will assume that the data bases are fully consistent; that is, one of the above options will be implemented to resolve any inconsistencies. The set of consistent data will be termed as the "scenario" for a given run of the integrated OCA system.

5. HIERARCHICAL SYSTEMS: THEORY AND DESIGN

5.1 A GENERALIZED THEORY OF HIERARCHICAL SYSTEMS

5.1.1 The Need for a Theory of Design of Hierarchical Systems

One of the goals of this project has been to determine the extent to which the concept of hierarchically structured information helps to manage the OCA mission planning process. The kernel aids themselves are obviously pieces of some ´implied´ hierarchical scheme. Yet it has been an ongoing task to identify just what that scheme might be.

We have been motivated to produce an integrated planning system. In the process, we have been led to explore not only connectivity, both necessary and desirable, of the given aids, but also the underlying principles of hierarchies that are required to ground each system-building decision. The intent of this section is to formulate the principles that we´ve identified into a rudimentary theory. Later we will use this theory as a context within which system-building factors are understood and system-building decisions are made.

The concept of "hierarchy" can be applied both to software systems used by humans and to human organizations. In both senses the underlying "hierarchicalness" seems to arise from the manner in which information is controlled, processed, and exchanged. This is not to say that organizational hierarchies can be mapped directly onto computational hierarchies. In fact, it is a major theme of this report and the goal of this chapter to make a distinction between the two and then to build a theory to support the design of hierarchical systems. The next section explains the distinction, first informally and then from a somewhat formal (computer-science) point of view. Section 5.1.3 then introduces the basic concepts of hierarchical system design, as we perceive them.

5.1.2 Organizational vs. Computational Hierarchies

It is common to think of a hierarchy as an ordered chain of entities/processes, or perhaps as a tree of processes/events. Each node in the chain tends to speak only to its 'superior and 'successor' nodes. In human organizations this often makes sense - privates rarely need to communicate with generals directly. However, this view of hierarchicalness is too rigid when our goal is to build a tool/system to support the organization and its planning. The private occasionally has a better picture of something that is of interest to the general; similarly (and far from occasionally) a low-level software tool may discover something that is important to the functioning of a tool designed for a very different part of the hierarchy of tasks. It is desirable for the low-level tool to be able to communicate its result directly and in a suitable format to all nodes that would be interested in that result, rather than 'up' through intermediate levels. Thus we make the observation that for integrated software suites it makes sense to establish lines of communication across many levels.

For example, the route-planning tool could determine that a particular target demands far too many resources, and hence is labeled "inaccessible." We'd like to provide a mechanism whereby the target prioritizer and the resource allocator and all other tools would suspend interest in that target until such time as it became accessible. In the best case, the route planner could form and in particular communicate this judgement to the rest of the system before other nodes spent much or any time on it. Note that this communication, if effected, would enhance the overall efficiency of the system (by minimizing excess planning or replanning at other nodes). In turn, the quality of the plans should increase due to the longer planning time available for the remaining targets; hence resource utilization should improve. (This example is motivated by the fact that targets have been chosen by planners in real-world conflicts and resources have been allocated, only to find that the target is rejected at the wing level. Often it will be too late to replan well using the resources thereby made available.)

We are now ready to state the distinction between organizational and computational hierarchies in a crisp way: Hierarchical organizations pass strong control rules down through the echelons of command, while leaving a relatively larger portion of the local management of tasks to occur without too much supervision. This is reasonable since the local agents are human/intelligent. In contrast, hierarchical software systems inherit the rigid task structure that is inherent in computer programs; many of the tasks or tools cannot under any circumstances be construed as intelligent. The result for software hierarchies is that a great deal of local supervision is required to ensure teamwork while global supervision has to be very flexible to account for the desires of many different users. Thus when we seek to define a theory of software hierarchies we will claim that what we are really after is a new set of principles of communication.

5.1.3 A Theory of Design of Hierarchical Software Systems

With the previous introduction and distinction as our starting point, we observe that hierarchically structured information is characterized along three dimensions. First, there is a dimension of ordinality: hierarchical information is arranged in a spectrum that is marked at one extreme by specific instances and at the other by general concepts. Any given ´node´ along this dimension could represent a single datum or, more generally, a subsystem that is responsible for a given task. The second dimension is the scope of a given part of the overall system. This can be taken as a representation of the size and nature of the task at one of the nodes just mentioned. Finally, there is the connectivity of the system. If we again refer to the use of nodes in the system, then connectivity is the RELATION between any pair of nodes. The connectivity of the system is the set of all node-to-node connectivity relations. Connectivity refers not only to the links between nodes but also to the communication that is permitted over that channel. The decisions regarding the ordering and the scope of the nodes will often be either natural or subject to a number of equivalent choices - choices which do not materially affect the behavior of the system. It is the commitments to particular connectivity relations (channels of communication) that will serve most to specify the system´s design and resulting behavior.

The concepts of ordinality, scope, and connectivity as we have defined them are meant to generalize chains and simple trees to more complicated structures - essentially, graph-like networks. In particular, it should now be clear that ordinality really denotes the position of a node (a node is any task-oriented structure) in a network of other nodes. In the limit, ordinality is a relation that is specified by the structure of the graph. (Most systems will tend to be organized along the lines of general principles proceeding to specific instances, and so the ordinality of the graph can still be usefully thought of in terms of the idea of the "spectrum".) The scope of a node is the simplest aspect of the graph/network; it is just the task that is performed at the node.

In one sense, this concept arises simply because the tasks must be characterized. But the notion of "scope" has the virtue that it frees the designer from a rigid view of the tasks to be performed. Hence we could speak about the evolution of aids (or a decision to use only part of the capabilities of an aid) as an instance of "variable scoping". Finally, the concept of connectivity frees us from simplistic views of input-output relations (such as ´mapping´ the output of one system onto the input of another system). For example, if the human planner is using OCAP then the man-machine interface can be characterized as a connection between a node or nodes in the system and an external node - the display device (or even between OCAP and the human). Thus connectivity can subsume man-machine interfaces within the hierarchical theory.

Once we have accepted this generalization we realize that we get a big conceptual bonus for free: we have a framework in which it makes sense to give each tool (each node) a high degree of autonomy and thus view each of the nodes as a sort of expert. In other words, we have not only ´outgrown´ the chain-like structure of conventional hierarchies but also the chain-like order of processing. The route-planning example was meant to suggest this possibility by having the route planning occur early enough that higher-level but essentially independent planning could profit from the information (as the TACC commander was not able to when the information had to pass down to the wing and then back up the line of command). Independence is obviously the key determinant of the degree of autonomy of a node. Available processing tools and processing power are also important considerations.

There are two major consequences when we build a hierarchical system with autonomous agents. First, it is possible to decentralize the planning process. Since our problem contains different aids on different machines and written in different languages, this view may prove to be very valuable. Second, different architectures are suggested by message-passing graph-like networks of experts than are suggested by nested chains of tools. In particular, the standard notion of a linear control flow that supports a routine calling sequence would be replaced by a blackboard system (or set of such) that manages multiple simultaneous processes and the inter-nodal communications.

This concludes our generalization of the standard notion of hierarchical systems. We observe that standard chain-like hierarchies are both a valuable subset of the generalized version just presented and a likely developmental path for some systems that will eventually live in the richer environment just described. If the goal system is essentially chain-like but uses a few extra links, then it would be natural to follow a path of first establishing the basic chain. However, if the goal system is intended to be decentralized, exist in a blackboard framework, or have a high degree of inter-nodal communication, then it makes more sense to begin development in this direction. We believe the latter to be the case for OCAP; this means that it will take longer to produce initial system behavior, but that the eventual system will be much more powerful.

The next section, 5.2, integrates these ideas with other factors that must be addressed to produce an overall design concept - such as database issues, users, the problem of evolving aids, and man-machine interface requirements. The broad view of communication amongst nodes just presented forms the underlying context within which the other issues are explored and interpreted. Then in section 5.3 we propose candidate groups of aids and note some of the aspects of the aids that suggest communication links be established. Section 6 continues this process for one of the groups of aids (TPA, KNOBS, RPA) and extends it into a system design.

5.2 DESIGN FEATURES FOR HIERARCHICAL SYSTEMS

5.2.1 Introduction to Design Issues

There are a number of issues that must be faced by designers of hierarchical planning systems. Among these are database issues, the functions to be performed, the basic frame of reference of the planner (we will call this his mode) that is using OCAP, the presence of interfaces between aids, a realm of user-oriented issues (from access to man-machine considerations), and in our case flexibility with respect to evolution of tools. A particular design arises from the nature of the tools themselves and the commitments the designer makes to each of these issues.

In the following sections we will discuss each of these issues. The theory of hierarchical systems, presented in the previous section, is meant to be a foundation within which each of these issues can be talked about and characterized. As such we plan to use the notions of ordinality (position in the hierarchy), scope (description of the task performed at the node), and connectivity (description of the form and permissible content communicated between nodes). It is our hope that this will result in a uniform treatment of each of the above issues. We also hope that this discussion will support a deeper understanding of hierarchical systems.

A final note about this section - the discussions that follow of each of the issues, when taken together, are intended to bridge the gap between the theory and the design. Hence in Sections 5.3 and 6 we will make reference to the theory but our primary emphasis will be on the tools themselves. Thus we will describe a design from the point of view of the details of the aids and not as elements of a network founded on certain communication relations. We have further tried to smooth the transition from theory to practice by exactly matching each subsection in this section to a corresponding subsection in section 6.

5.2.2 Data Base Design

In this section we will review some of the salient points of chapters 3 and 4. Our goal will then be to abstract some of the issues; some will turn out to resemble issues that the hierarchical theory addresses. We will thus sketch a data base theory that is analogous to the hierarchical theory (with the hope that a data base so structured might better support a hierarchical system).

In chapter 3 we described the data bases used by each aid. We noted several different languages, the differing formats adopted (boolean variables, record-structures, real numbers, etc.), and the machine and/or system that the data base resides on. At the level of the individual aid, we are entitled to think of the data base as having a sort of physical integrity - each aid has its data base, or system of such, and that data base is a unit. However, as soon as we began to look across the different data bases (chapter 4) we found that we cannot think of the data bases as inviolable units.

We observed two types of differences amongst the aids' data bases. The first type retains the physical character: machine X versus machine Y, records on X written in C versus frames on Y written in LISP. The second (non-physical) type of difference is the CONTENT, the meaning, of the data base entries. In chapter 4 we addressed the need for a mechanism (the consistency manager) to ensure that the data bases remained consistent with each other. Once this manager is built the data bases are no longer "inviolable units"; they are an integrated and functioning whole.

The problem of maintaining consistency is difficult. We observe that the physical aspects of the different data bases suggest the nodes that we've seen in our hierarchical theory. Also, the semantic aspect (the information content) of each data base and the (informational) relationships across data bases suggest the hierarchical notions we presented. Recall that each data base has aid-specific information as well as information that is also of interest to another aid. It's easy to imagine the information of each aid as being partitioned according to whether it was specific to that aid or a piece of 'common wisdom'. Each of the aid-specific parts of the data bases resembles an autonomous agent, responsible for its own integrity in precisely the manner that independent functions within aids were autonomous and responsible for a given task. We can pursue the analogy - the communication of consistency information certainly resembles the communication of information amongst nodes. We used the idea of ordinality to capture the ultimate task structure of the hierarchical process; this allowed various processes to run at once, and led to blackboard architectures supporting inter-nodal communication. In particular, ordinality lends itself to diversity and to somewhat amorphous task structures (ie beyond input/output chains). Partitions across independent information along with a background of communication of consistency procedures or data strongly suggest that the ordinality and connectivity concepts have natural analogues in data base design. (For example, the ordinality of the hierarchy becomes a generalized partition function on the data base.)

We can sum up the last by saying that data base analogues exist for each hierarchical notion, with the most important and interesting difference being that the transmission of consistency information may be very different than the communications within the aids. (This may not be true of interfaces which are designed to mediate inter-model inconsistencies. See Section 5.2.5.) The data base design begins to resemble a network in miniature; this leads to an important design principle. This is, simply stated, that we can imagine designing the structure and connectivity of the data base with the structure/ordinality and connectivity of OCAP in mind. The basic idea is to tailor the structure of the data base to that of OCAP: for example, we can imagine mappings between nodes in OCAP and the analogue of a node (such as one of the aid-specific partitions) in the data base system. Figure 5-1 is an attempt to present this idea visually; each task has a direct pipeline to the data it requires, with the result that the overall data base may exhibit, albeit implicitly, a hierarchical structure to match that of OCAP. Links between data nodes represent consistency channels; those between system nodes represent any of several types of inter-task communications.

We note in passing that the consistency manager described in chapter 4 is itself hierarchical. At the top level the broad context defines the consistency rules, which then run at the lower level across various parts of the data bases. This would be a natural place for an explicit hierarchical characterization of the data base system.

This concludes our abstraction of the design of the data base system. The hierarchical concepts serve not only as metaphors for the consistency manager of chapter 4 but also as a framework within which that manager can be defined and within which different data base designs can be rigorously compared. The remaining sections will focus on OCAP, yet we will always attempt to use the theory as we have presented it here - to identify principles that enable us to compare alternatives and avoid purely ad hoc designs.

5.2.3 Required Functionality

This section discusses the requirements for a hierarchical system such as OCAP. We will begin the design with a hierarchically structured organization and with a pre-ordained set of tools to be placed in the eventual design. In Chapter 1 we remarked that a hierarchical

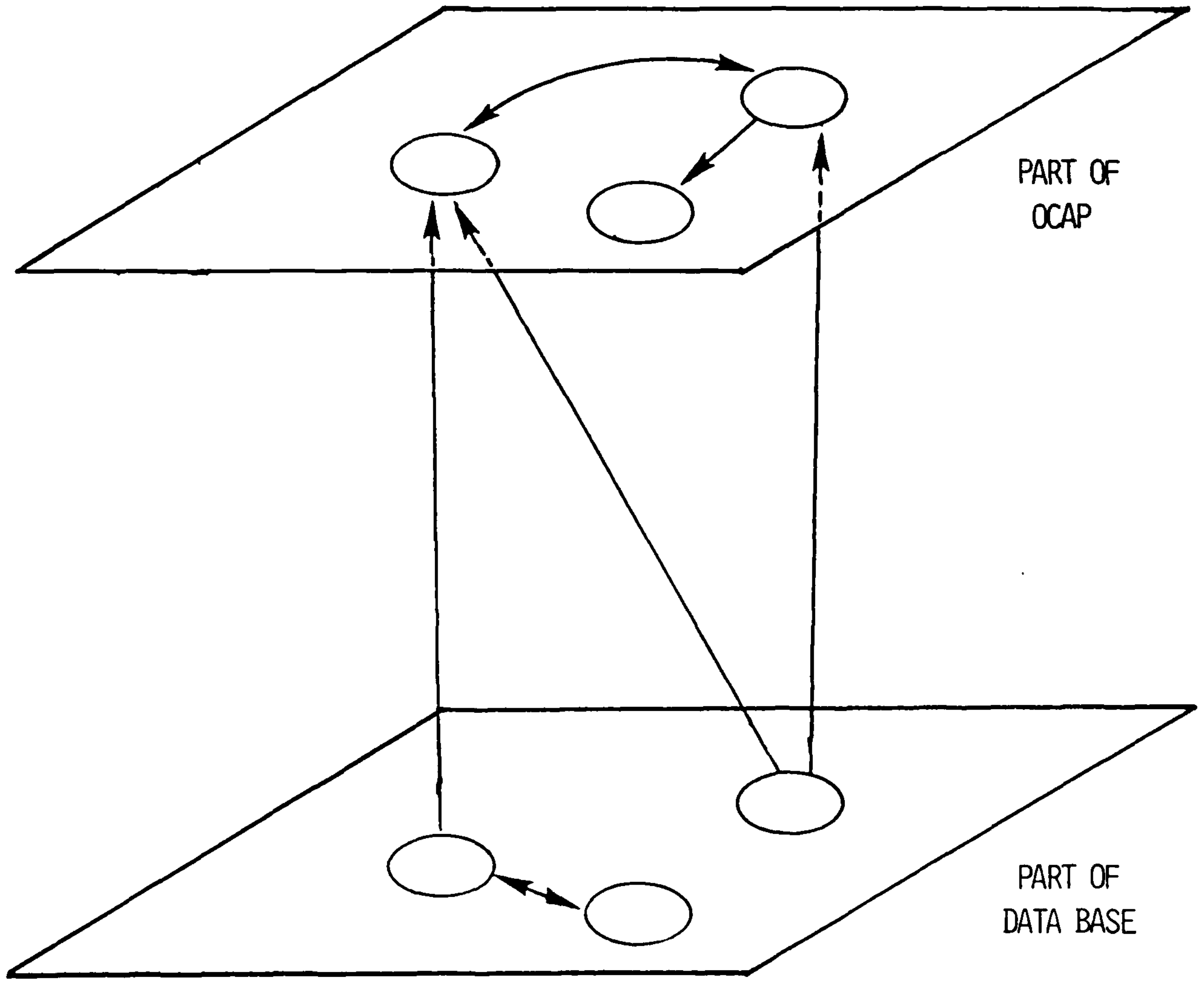


Figure 5-1: Task-Structured Data Base Design

organization and a hierarchical software system are not "hierarchical" in the same sense.[1] This fact is the source of our remark in Section 5.1 that there is a distinction between hierarchical organizations and hierarchical software tools that are meant to support the planning of those organizations. This section will characterize the tasks of hierarchical designs in such a way that we can account for both of these ´constraints´ on what is possible and desirable for the OCAP.

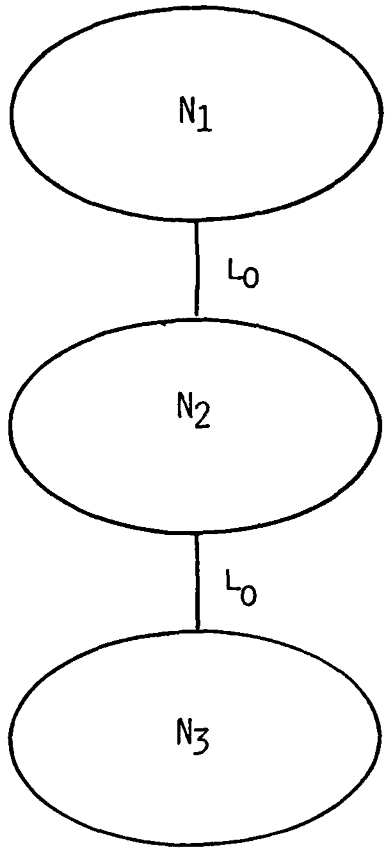
A top-down view of system design emphasizes the tasks that must be accomplished. This view takes its cues from our theory and from the organization the design is meant to support. In parallel with this there is necessarily a bottom-up view, which is motivated by the particular software tools that we possess. We will attempt to characterize the interplay of these two design foci according to a MAPPING between abstract tasks and available software. This leads to an iterative procedure; each view is refined by the other. In this way we hope that the design of all of the new software and especially of the communication links will seem natural. Actually, the result will be "natural" only after a sufficient number of iterations.

Functional requirements for the design are derived from system concepts, organizational structure, and output requirements. Conceptually, we have nodes that perform tasks and are defined in terms of ordinality, scope, and connectivity relations. These concepts are so general though that they do not initially specify much about the design. The initial system design is thus a result of the organizational context and requirements.

1 We quote: "Hierarchical organizations pass strong control rules down through the echelons of command, while leaving a relatively larger portion of the local management of tasks to occur without too much supervision. ... In contrast, hierarchical software systems inherit the rigid task structure that is inherent in computer programs; the result for software hierarchies is that a great deal of local supervision is required to ensure teamwork while global supervision has to be very flexible to account for the desires of many different users."

There are two senses in which the TACC can be characterized as hierarchical. It is hierarchical in the sense that there are levels of command, with any given level talking primarily to the levels just above and just below. It is also hierarchical in the sense of the fairly strict sequence of planning actions that occur: intelligence analysis is ongoing yet precedes target selection; this precedes resource planning, which in turn precedes path planning. This organizational structure supports a number of easily identifiable tasks; target selection, resource allocation, and path planning could form our original requirement specification. The communication between the nodes, if not specified now, leads to an abstract map of the system, such as Figure 5-2, which shows the initial nodes and makes no distinction between the types of communications. To refine this design, we can specify the communication that is implied for each link, or we can overlay what we have in the figure with the actual aids and use this to drive the first refinement.

Choosing to follow the bottom-up approach we immediately identify TPA with target selection, KNOBS with resource management, and RPA with path planning. This imposes a great deal of additional structure on our graph. (Remember that we are looking for a new abstract graph, so that our refined graph DOES NOT contain a TPA, KNOBS, or RPA node. The graph is an abstraction of the task structure; the functions in the available aids are always defined as a relation on this network. This is the sense of the ´mapping´.) We require input nodes, and the communication that passes between the input nodes and the task-oriented nodes is fairly straightforward. We also see that target selection should be broken out into sub-nodes which reflect target identification and target nomination. Target nomination in turn responds to resource availability and a target prioritization task. This is where it gets interesting; on the face of it it seems that we can equate the resource availability node introduced for TPA with a resource node in the resource allocation task - let´s call this portion of the resource allocation node a current-resource-maintenance node. We will refer to the remainder of resource allocation task, after current-resource-maintenance is removed, as resource-distribution. The design, as we have specified it thus far,



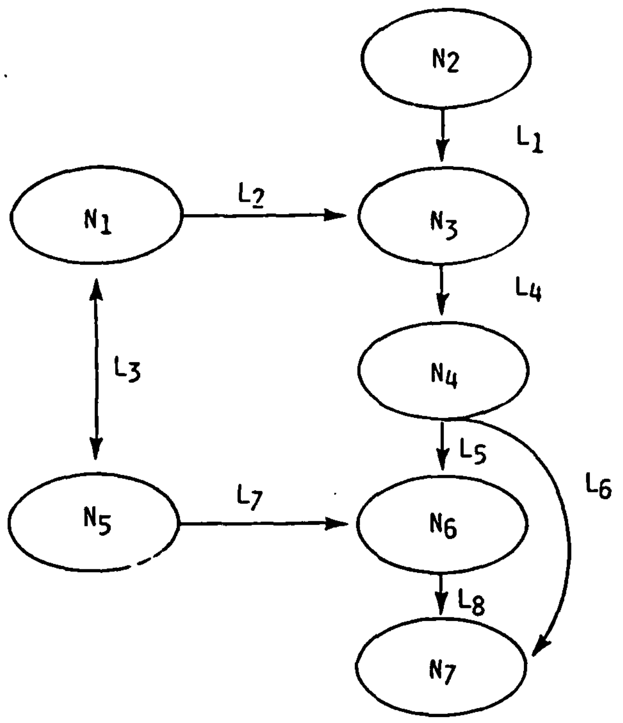
NODE N_1	:	TARGET SELECTION
NODE N_2	:	RESOURCE ALLOCATION
NODE N_3	:	PATH PLANNING
LINK L_0	:	COMMUNICATION LINK; FORMAT, DIRECTION, AND CONTENT UNSPECIFIED

Figure 5-2: Initial Design Network

has the structure of Figure 5-3. Note that the link specifications now have a directionality and a specified content. These specifications are provisional, especially since the format has not yet been specified.

We don't need to continue this particular refinement sequence down into the resource and path nodes too much more; we are ready for the point of the exercise. This is that the definition of the communication links, which attends the definition of nodes, often turns out to be the primary motivation for a design modification. This occurs when we compare the new links with the existing aids according to our mapping. In Figure 5-3, there is the implicit presumption over the resource availability channel that all forms of resources of interest to the target prioritization task are passed. Thus we expect the number of aircraft at each base and of each type are passed. At the next iteration, when we associate the revised task structure with the actual aids we run into a problem: TPA only plans relative to F-111's. This suggests a major design decision must be made (we finally get to the point of our example).

In terms of our theory, the design decision is rendered as a choice between two options. The first option is to modify the scope of the resource allocation node - that is, specialize it to F-111's only. [This option is in fact mentioned, in two contexts, in Section 6. We refer to this as an 'F-111 Planner'.] The second option is to add non-F-111 planning capability to the system. The second option can be viewed two ways: as a modification of each of the nodes in the target prioritization task (i.e. all nodes broken out of that task) or as an additional node, to be added to the network and to communicate not only with the target prioritization task nodes but also with the current-resources node. The second option enhances the capability of the system over the capabilities inherent in TPA - the first view does this by improving TPA; the second view does so by adding a new capability outside of TPA that becomes part of the overall design. We will favor adding a capability outside of TPA, that is a new task/node in the system, rather than attempting to modify TPA itself. We will refer to



NODE	N_1 :	RESOURCE - AVAILABILITY
	N_2 :	TARGET - IDENTIFICATION
	N_3 :	TARGET - PRIORITIZATION
	N_4 :	TARGET - NOMINATION
	N_5 :	CURRENT - RESOURCE - MAINTENANCE
	N_6 :	RESOURCE - DISTRIBUTION
	N_7 :	PATH - PLANNING TASK
LINK	L_1 :	TARGET LIST
	L_2 :	RESOURCES FOR EACH CATEGORY
	L_3 :	BIDIRECTIONAL EQUALITY CHECK, SEMANTIC MATCH, (PROVISIONAL)
	L_4 :	SORTED TARGET LIST
	L_5 :	TARGET LIST + EXTRA INFO., FORMATTED FOR ALLOCATION ALGORITHM
	L_6 :	TARGET LIST + EXTRA INFO., FORMATTED FOR PATH - PLANNING ALGORITHM
	L_7 :	RESOURCES FOR EACH CATEGORY, (PROVISIONAL)
	L_8 :	RESOURCE - TARGET PAIRS

Figure 5-3: The Next-Generation Network

these extra nodes as interfaces - for obvious reasons. The idea of an interface will be formally developed a bit further in Section 5.2.5. Actual interfaces that we have designed are discussed in Section 6.

In addition to creating nodes to fill in the task gaps that arise, it is easy to see that there may be functions in the aids that the abstract network never calls upon. For example, a duplicated capability might correspond to a single node and a single connectivity relation - in which case one of the functions will lay dormant. (The connectivity relation here resembles a "quotient function" in topology.)

We mentioned that the communication links tend to drive the design decisions. Another aspect of this (besides the promotion of interface design) is in terms of the summaries that might be expected of a particular node´s results. Often these are the result of the organizational context upon the network that is being designed. (We could formalize the relation between the organizational structure and the structure of the abstract network, but this is unnecessarily formal for our present purposes. A simple example, which will come up a number of times in the sequel, is that of summaries and abstractions. Summaries refer to a tailored output of data for the use of a particular type of user. (For example, a prioritized target list could be passed to the chief of combat plans for review with all of the rest of TPA´s output removed.) An abstraction is very similar to a summary - it carries all of the relevant planning information for a miniature planning task to be initiated. (Thus if one wanted to set up a path-planning run with modified defense installations an abstracted test case could be set up and shipped off (communicated to) the path-planning nodes. While both of these forms of communication have superficial similarities, they actually represent different types of communication links. This will be discussed further in section 5.2.6.

The basic elements of our design program can now be stated: propose an initial network consisting of task nodes which satisfy the basic system requirements. At successive steps, revise this abstract/design network of tasks and communication links. The actual aids and organizational requirements will each force modifications upon the design network. The goal is to produce a fully integrated design in which the mappings that we have spoken of all faithfully connect task structures with actual capabilities or proposed additional capabilities.

We believe that the final system is "complete" in a sense that can be made rigorous. In fact we can now state that the reason for working with the abstract design system and framing correspondence mappings, rather than working with the actual aids and looking for a design that fits everything together, can be stated (informally) in terms of completeness. The representational formalism that we've introduced seems to be powerful enough to completely characterize the requirements and behavior of a hierarchical system. Thus if we begin and stay within the purview of the hierarchical concepts we should produce a design that is complete in its tasks. (Admittedly this is a hypothesis, not a proven fact, at the present time.)

The mapping from the design network to the existing capabilities has been sketched intuitively thusfar; essentially, we have not characterized the mapping so much as we have pointed to its existence. We will not extend the idea into a fully-specified mathematical framework in this report - though intuitive use of the ideas of domain, range, and mapping properties will occur. We will wonder particularly about the NATURE of this mapping.

The next section will show how the mapping leads to an important view about the semantics of the design network and thus the behavior of the eventual system, OCAP. We will follow this discussion with an elaboration of two issues that arose in this section - the need for interfaces and the many types of communication links that must be designed.

5.2.4 The System Mapping and Mode-Based Planning

In the previous section we introduced the mapping from our abstract network to the existing aids. We will refer to this mapping as the "system mapping" from now on. In this section we want to sharpen this notion of the system mapping. This will lead us to the concept of "mode-based planning".

One view of the system mapping is that it measures the degree of correspondence between the current, proposed design network and the tasks that the existing software performs. Within this view it isn´t important which entity is primary and which stands in relation to the other. This view is useful at the outset (the first few iterations of the design methodology introduced in section 5.2.3); it is a way of collecting the broad system requirements and the available technologies and mediating their differences. At a certain point however, we will want to emphasize the network we are designing over the existing software. For example, this will happen when we reach the point of deciding whether to use, modify, replace, or support a given existing capability. (Recall the example in section 5.2.3 that led to a decision between scope modification and interface design.) Furthermore, we will define operators on the network, not on the existing aids´ or on their tasks. Thus we are led away from a correspondence measure and toward a true mapping: the system mapping has the design network as its ´domain´ and the existing tasks or the aids as its ´range´. Note that the operators implied by using, modifying, replacing, and supporting have the space of networks as both domain and range.

The design network must support user needs in addition to accomplishing certain tasks. Some of these needs will not correspond to any available software. For example, it is obvious that none of the current aids has software that can produce a summary of the entire plan. This encapsulation requirement arises only in the context of the system as a whole. For this reason we will broaden our concept of the system mapping to include addition requirements (such as man-machine needs) in the range of the mapping.

Another view of the system mapping, which lies at the heart of mode-based planning, is that it represents the semantics of the network. The network is an abstraction, a set of symbols. The system map is used to say that the symbols correspond to behaviors that we are familiar with (current tasks performed) or to additional needs (man-machine for instance) that we can envision. Thus the mapping is the denotation of our network; it is the relation between a symbol representing a node or a communication link and the MEANING of that node or link.

This sketch of the system mapping leads us to the question: "Is there more than one system mapping, and if so can they exist in the context of the same design?". The answer is that there is an important role for different system mappings and that a good design should support several at once.

We've said that the system mapping specifies the semantics of our network. If we imagine different mappings being applied to the same network then we will have introduced different interpretations of the same task - be it a problem solved at a node or data communicated in some format over a communication channel. We will call these different interpretations different "modes".

It is easy to see that mode-based reasoning is something we humans do all of the time. If we are an expert, or a high-ranking official, or pressed for time, we will tend to view a given requirement (say to create a plan to attack a region and then to describe that plan) in different terms than if we are a novice, or a junior staffer, or have all the time in the world. A mode is thus a context that defines our interpretation of a given task. Our design must recognize that there are different users, each with their own interpretations, and there are even variable contexts for a given user. Our design must support these interpretations; it will do so by allowing different planning modes.

We will represent each mode as a system mapping. We will support different planning methodologies, different users, and individual users who change their mode (detailed-study to quick-overview) by pre-designing different modes. We will then coordinate the application of all of the current mappings and update the design. Often the additional mappings will simply pose extra requirements - though this will not always be so. The most interesting cases are precisely those in which a given element in the network has a different interpretation depending on the mode. This happens quite often for the communication links: when a user initiates a planning session a context is loaded (depending on the user type and type of planning the user decides to engage in - such as actual planning vs. what-if games) and according to this context the output transmitted from node to node and the output transmitted to the planner will be defined. For example, an officer in the WOC would get detailed flight path information from the path planning node, whereas a colonel that is in charge of the WOC and responsible for generating a message to the TACC might (from the same node!) simply get a list of feasible and infeasible targets plus a line of explanation for each infeasible target.

The overall system is designed to support variable system mappings. Extra nodes and links are created as required, and context dependent tasks and connectivity relations will be generalized so that they carry along a ´tag´ that reflects the context. When a system mapping is applied to the (one) design network the given mapping will trigger all of the appropriate tags and the complete denotation of each task and communication then becomes available. (We can speak of this as the interpretation of the design network relative to a given mode.) In the same way, when the system is engaged a context is automatically set up. This context defines all of the functions (tasks and communication between task and from OCAP to the user). Before the system is engaged all of the functions are defined only to the point of being relative to a certain context. Thus the context (the mode and the user) has to be entered before OCAP is fully initialized.

Note that in some contexts certain nodes might be inaccessible - for example, suppose that the WOC user cannot change the list of permissible targets, or perhaps is denied access to the target prioritization nodes as a whole. Thus there is a sense in which the context will end up not only changing the meaning of some tasks and communication but also the structure of the network; in some contexts some nodes will be added and others removed.

The concept of a mode or of variable system mappings has far-reaching consequences, only one of which is the ability to characterize access privileges in a crisp way. We will not be able to pursue it more here, yet it appears that it could have a prominent effect on the types of design networks that are considered. However, in Section 6 we will mention three different modes of planning, each of which would correspond to a different system mapping that in turn identifies a different interpretation of the design network.

5.2.5 Design of Interfaces

In section 5.2.3 we found that there were tasks that the overall system had to accomplish that did not exist in the given aids. In general, there are two types of tasks that don´t come with the aids. These are mechanisms to permit the aids to communicate with one another and machinery to facilitate communication between the planner and the system. The first of these two is the set of interfaces required, and is the subject of this section. The second (man-machine nodes) are discussed in the next section.

Up to this point, a node has been characterized as performing a task, with the task defining the scope of the node. Since the nodes have solved domain-type problems it has appeared that the ´purpose´ of a node is to solve a military planning problem or subproblem. If we were to represent the domain problem in the form of a flowchart, then the nodes would each be structurally related to that flowchart - each node could be associated with a unique part of that flowchart.

Continuing with our flowchart analogy, we may find that some of the tasks defined by the domain problem simply didn't get solved in the existing aids and these will require new nodes. These nodes are similar to the domain-type nodes just mentioned; they solve a problem that is missing (in the aids) when we decompose the domain planning problem into tasks. Since this is an inter-aid problem we'll refer to the node we create as an interface node. For example, one aid may have a capability that is important (such as being able to plan for all types of aircraft, not just F-111's) and we will decide to augment the other aids rather than just use a portion of the available tasks from the given aid.

There is a stronger sense of interface-type nodes that motivates the name. The existing aids were designed independently and the result has been that they tend to make different assumptions about the world. And even when they make the same assumption about some aspect (such as an output of one aid that is used as an input to another) they will almost certainly have a different representation of that information. Thus interface-type nodes are designed to resolve inter-model inconsistencies of structural (i.e., mediating differential assumptions) and communicative (i.e., mediating differential representations) forms. Note that both the structural and communicative forms respond to the requirement to set up a certain kind of connectivity relation - one communicates and resolves different points of view, another translates from node to node.

Once the 'purpose' of a node is established (problem-solver, interface-type, etc.) we can set about designing the node itself. In this context of interface-type nodes special design issues arise; in the remainder of this section we will highlight these issues.

We would like the node's purpose to turn out to be the system mapping function (see sections 5.2.3 and 5.2.4) as applied to that node. But this is tricky because the existence of an interface node (an element of the domain of the system map) by its very nature implies a gap of one form or another in the aids themselves, the range of the system mapping, - so that if there is nothing to map to we have to wonder how we initially recognize the need for nodes in the domain. (We have said that some nodes arise out of organizational requirements of the design - yet these are unlikely by themselves to result in all of the necessary interfaces.) The natural approach seems to involve having the designer recognize gaps or defects in the range space and use them to construct desired connectivity relations in the domain. In other words, we do not recognize the need for nodes in the domain without first defining the gaps in the range (of the system mapping). (This is, after all, what we do informally: we see that an interface is needed, then we look to the communication requirements and build something to perform the necessary tasks.) From the abstract point of view this involves creating both a node and its surrounding connectivity relations at the same time (i.e., as one integrated process since it is the node <u>and</u> the communication that is required). There is probably a characterization in terms of the system mapping that would do this for us; we won't pursue it here though.

When we spoke of database issues, and even before when we originally addressed the general architecture within which a hierarchical system would be most 'comfortable' (the result was a blackboard architecture) we hinted that the objects communicated between nodes were not in any way inherently limited. In particular, the data that is passed will not always be a list or a table or other passive output-type object. In fact, we can turn this around and make the positive observation that in some cases a procedure will be the object passed. For example, the allocable resources that KNOBS knows about may not be equal to those initially assumed by TPA. In this case we found that a procedure was required to resolve the disparity. If we allow procedures to be passed then we can characterize iterations in the network and even permit nodes to bargain for resources (both in the domain and for computational resources within the system)! Thus we envision an active network of experts that passes not only formatted data but also procedures, requests, and perhaps even guidance.

There are a few low-level design issues having to do with where an interface node resides in the network. That is, there are instances in which the original specification of two graphs differs but the behavior of the two is identical. A simple example is the case where a node is broken down into two sub-nodes with a simple channel between (the original node did two independent tasks in succession). There is one instance of building an interface however in which the decision of what structure to adopt is very important. This is the instance in which the interface node is not pulled out and given its own status in the network but rather when the node is merged with an existing node. (In terms of our theory the first option adds new connectivity relations, ordinal relationships thus are added, and the only scoping issue is for the new node. However, the second option has no effect on the structure of the graph, possibly some on the connectivity relations, but will generally have a profound effect on the scope/task of the node that is merged to.) We encountered this problem when we initially thought about designing a TPA-KNOBS interface and a KNOBS-RPA interface. It occurred to us that we might use the rule-processing capability of KNOBS as an environment for the interface. The idea was to express all of the consistency requirements as rules (perhaps with appended procedures) and have the KNOBS rule interpreter now become an expert interface as well. Above all, this solution seems to be a very elegant use of KNOBS. In terms of design issues, which is our focus here, it remains an open issue and one with notable consequences for design. There are also important consequences for behavior - since a merged network and a non-merged network may end up differing dramatically. At any rate, there seem to be practical reasons for ruling out the ′KNOBS-based-expert-interface′ at the present time. The most prominent reasons are efficiency of the rule processor and the desire to be insensitive to replacement of aids over time. (We know that TEMPLAR′s rule processing capability will be quite different from that of KNOBS. Also, we expect path-planning technology to evolve rapidly as new weapons, sensor systems, and offensive capabilities evolve.) We would not rule out expert-system interfaces in general. The underlying point here is that the efficiency and insensitivity criteria are not met by an expert system which embodies multiple and disparate forms of expertise.

This completes the discussion of interfaces. They perform a special task, so it is not at all surprising that new design issues arise for them. These issues have only been touched upon here, but perhaps enough has been said to convince the reader that most of the design issues (and the theory that supports the design) are tested more in the context of interfaces than in any other single area we've discussed.[1]

We will proceed now to discuss the second case in which requirements are specified for the system which have little or no basis in the individual aids - the man-machine requirements. These requirements take various forms, but almost all stem from design goals that pertain to the whole, integrated aid rather than to its parts.

5.2.6 Design of Control and Man-Machine Interactions

There are many types of communication schemes present in a system design; we have already mentioned a number of them in this report. For example, an overall control of procedures is required, as well as a mechanism that controls access to procedures based on user privileges. Summaries at various levels and of variable sets of processes are desirable, both for human planners and for setting up test cases to

1 Of course, if our basic design metaphor had always been to simply connect the existing aids in some suitable fashion then it comes as no surprise that all of the action exists in the design of the interfaces.

evaluate a given aspect of a plan (we called these abstraction cases). Problem-solving nodes communicate when their task is ready to begin and when it is completed. Finally, there is communication amongst these nodes in the form of guidance; we gave the example of the path-planning expert notifying the rest of the experts that a given target is and will remain inaccessible for a specified period of time.

There are really only two types of communication occurring here. First, we have interactions amongst sets of nodes that solve problems (the ´experts´). This is just the internal or automatic control of the system. Second, we have interaction between sets of nodes and the human planner; this covers all of the man-machine interactions.

We conclude from this typology that there are two forms of connectivity relations in the system, depending on whether a human is involved or not. We immediately propose that the concept of connectivity be specialized to account for these two forms alone. This has a large impact on our concept of design - since only two types of representations of knowledge will be allowed. In effect, there is an implied design principle that says that a single representation of inter-nodal communication and a single representation of human-nodal communication is required.

One way to implement this idea is to use a blackboard architecture and to design the protocols that manage the blackboard either to ´talk´ to other parts of the blackboard or to talk to the human planner. This sort of architecture, by enforcing uniform communication between tasks, implicitly forces some degree of uniformity on the scope of the tasks at each node. Thus the system will tend to be of even granularity. This contrasts typical design approaches - in which each type of communication may have its own characteristic format and individual tasks may be large or small. Finally, the architecture based on these two languages is not sensitive to modification and replacement of the aids; it is easy to support. The primary drawback is that this approach requires a higher initial investment of effort - to design the languages and fit the existing communication within these languages.

5.2.7 Evolutionary Considerations

The kernel aids are not rigidly defined for our system. Each of our design principles must recognize that the aids may evolve and/or be replaced over time. Similarly, we expect that new aids solving new tasks will appear and we must be able to accept them too. Hence the design ´principles´ of previous sections, as they are refined in the course of designing OCAP, must remain flexible.

5.2.8 Summary of Design Issues

The purpose of this Section (5.2) has been to produce a set of design rules from our theory of hierarchical software systems. These rules are meant to formalize the use of that theory - they do not constitute a design themselves.

We would like to produce a next-level analysis of the trade-offs between design principles. For example, we do not want a concern about flexible support of evolving aids to deter us in any way from producing a working design and implementing it in a reasonable amount of time. Such a theory might bridge the gap between our theory and design principles and our design task. We have decided, however, that the study of the trade-offs is best accomplished by actually designing OCAP (at least an initial design); an abstract theory of trade-offs doesn´t appear to be as valuable. Thus we will shift gears now and begin actually designing hierarchical systems; we will conclude with a design of OCAP, in Section 6.

5.3 CANDIDATES FOR SYSTEM DESIGN

In this section, we present possible candidate sets of aids for integration. A goal of this project has been to identify design principles as well as to present actual architectures for design. Thus, we will first present our basis (criteria) for choosing particular sets of aids for integration. Then we will present the candidate sets and evaluate them according to the criteria.

5.3.1 Criteria for Choosing Candidate Sets

This section provides a set of four criteria for choosing the candidate sets of aids for integration. These criteria provide a set of positive specifications for candidate sets, and also a way of pruning out those candidates that are not valuable for several reasons. The evaluation of candidate sets of aids in terms of the criteria will be subjective in nature.

These four criteria are (1) the completeness of the task that the integrated aid addresses, (2) the extent to which a particular individual (job type) is likely to profit from the integrated aid, (3) the degree to which the new aid is hierarchical, and (4) a list of pragmatic issues. Each of these is described below.

The completeness of the candidate set, is defined in terms of the meaning of the task that the proposed integrated aid supports. In our theory of hierarchical systems we have identified the task that is performed at a node as the scope of that node. If we were to "draw a node around all of the original nodes" (we'll leave this vague) then we could consider the overall aid to be a "super-node." Then the scope of the integrated aid is the scope of this super-node. In particular, we can simply look at the remaining connectivity relations (in this case.

primarily the range of admissible input queries and the man-machine interface) to determine whether a complete set of issues is addressed. With this framework we can speak at least semi-rigorously about ′gaps′ between existing connectivity relations, and we can evaluate the results that aren′t being presented. This is the most important criterion.

The second criterion, the degree to which the proposed candidate set will support an individual, is more difficult to formalize. The basic idea is clear though: a particular group of aids may provide "complete" support for a given overall task - yet if there is not an individual that would profit sufficiently by having this task solved then the quality of the integration (as measured by the completeness) is irrelevant. The task is irrelevant if we cannot establish that a user should be performing it, regardless of whether such a job currently exists, or may exist in future planning environments. Thus, the difficulty with formalizing this criterion is that we are required to evaluate not only current job roles but also proposed job roles. This criterion is important, and it should not be finally evaluated without the joint input of the Air Force planners and the technologists. For the purposes of this project, though, we applied it based on our own understanding of these viewpoints.

The third criterion is that the proposed aid should be hierarchical. This is important since the effort of this work is not only to produce a particular *tool* but *also* to investigate hierarchical systems. Obviously this criterion is motivated not so much by a broad perspective as it is by a project-oriented view.

The final criterion is that the proposed candidate set should be feasible to build, test, and evaluate. There are several pragmatic issues here. Among them are the scope of the candidates themselves as they stand, the desirability and feasibility of creating nodes that reflect a subset of a given aid′s functions (eg, dropping out RPA′s explanation facility), and the data base problems. We will also have to face the effort involved in creating a demonstrable system both in terms of the functions finally settled upon in the design, and the host of hardware problems to be solved.

The evaluation of several candidate sets using these four criteria occurs in the following sections. Candidate sets that failed several of the criteria are not mentioned. For example, it is hard to see how a CTA-KNOBS aid a) solves a complete problem, b) defines a user who would profit enough by having it to merit the cost of producing it, or c) establishes a hierarchical system.

5.3.2 Description of Candidates

We will describe four candidate sets of aids which most closely matched the set of criteria described in the previous subsection. The first subset (CTA, DART, DAGR, TPA) represents an OCA planning tool that can be used in the Combat Operations Intelligence Division (COID) in a TACC. The second subset (TPA, RPA) is a first cut at a hierarchical system. In the third, we add KNOBS to the second group to design a system that can be used across three levels of the planning hierarchy. Finally, we add DAGR to the third group to augment the data base management at the highest planning level (TPA level).

5.3.2.1 Intelligence Tool (CTA, DART, DAGR, TPA)

This candidate architecture can be designed as a tool for the Intelligence level of the planning hierarchy. CTA and DART work together to assist the C^3CM Analyst determine which C^3 nodes are the most important to attack. DAGR and TPA assist personnel in the COID to determine which airfield targets to attack. Together, they help build and prioritize the list of candidate OCA targets.

CTA-DART

CTA and DART were individually designed to assist the C^3CM Analyst of the TACC. CTA allows the user to consider different generic strategy objectives such as remove command, falsify information, jam, etc. and select those actions which maximize the probability that a particular desired outcome (Deny Command, Decrease Control, etc.) will be achieved. DART, on the other hand, is more of a data base manager. It allows the C^3CM Analyst to identify critical C^3 nodes in real time.

To use these aids in an integrated way, the user would run DART to update the C^3 node data base, based on incoming messages and DART's analysis of them. He would run CTA, in sequence or in parallel, and plan the best set of strategy objectives and evaluate these objectives based on cost and benefit. The user would use the up-to-date data base produced by DART and these strategy objectives proposed by CTA to help him determine which C^3 nodes to attack. The integrated aid itself cannot make the decision for the C^3CM analyst, but it would provide him with valuable information to make a reasonable choice.

Note that there are no major interface problems between CTA and DART. They use separate and distinct data bases; thus, there is no consistency problem. There will be no direct communication link between CTA and DART. An interface is needed simply to ensure that DART is run often enough to provide an up-to-date data base at the time CTA is used. The interface also will lead the user through a systematic use of both aids.

DAGR-TPA

Both DAGR and TPA help the personnel in the COID develop an Air Order of Battle (AOB) file and determine which targets to attack.

DAGR and TPA lend themselves naturally to integration. TPA uses, as input, the AOB file to determine the best set of airfields to attack. DAGR updates the AOB based on Kill Reports and Movement Summary Reports. Thus, DAGR should be run before the user runs TPA, so that the most accurate information is available. Note that although DAGR updates the AOB, more inputs are needed to the AOB to reflect the true state of information on the AOB. DAGR is not a complete AOB-update tool.

Because both aids use the same information but the information is stored in separate data bases, this data must be kept consistent (see Section 4). An interface is necessary to produce any communication between the aids. (Actually, the communication is one-sided. When DAGR updates the AOB file, it must inform TPA of the changes. TPA has no reason to communicate information to DAGR.) In a more automated system, after the user logs onto the system, the interface will run DAGR, update TPA's data base accordingly, and allow the user to run TPA at that point. In a more human-guided system, the user would have to direct the interface to run DAGR at the appropriate time.

Combining All Four Aids

After running CTA-DART and DAGR-TPA, the user will have two lists of OCA targets: a list of critical C^3 nodes from DART and a prioritized list of airfields from TPA. There may be some overlap; some of the C^3 nodes may be on the airfields which TPA outputs. The user will have to coordinate the two lists to arrive at one set of targets that should be attacked.

The integration of these four aids does not allow for any systematic way of using the two outputs from DART and TPA mentioned above. Possibly, the user-interface could graphically display both the airfields and C^3 nodes. This display would help the user in deciding how to coordinate the targets.

This candidate architecture is not hierarchical. It is designed only as a tool for the COID of the TACC. Although the architecture provides the Intelligence Officer with valuable information, it is not a complete tool. The Intelligence Officer will need more information to make decisions than what is offered by these four aids. Thus, this candidate system fails at least two of the criteria, and we do not consider it valuable to build as a hierarchical OCA planning aid.

5.3.2.2 Two-Level Hierarchical System (TPA, RPA)

A very simple hierarchical design of an OCA planning system consists of only TPA and RPA. This system would use TPA to determine which targets to hit and would compute the best path to those targets using RPA. The system would be strictly an F-111 planner since both TPA and RPA were specifically designed for F-111′s. Any extension to non-F-111′s would require the addition of KNOBS (see next subsection).

The Target Nominations Officer in the COID would be the most likely user of the system. He would have access to the same data bases as the route planner at the WOC has. The WOC planner would certainly want to use the RPA-module of this system, but would probably have no reason to use the TPA-module.

The user of this simple hierarchical system could use the tool in at least two ways. By running RPA first, the tool could compute the survival to each target and use this number to adjust TPA′s cost. (If the probability of survival to a particular target is very low, then the cost of getting there should be much higher than if the route were safer.) Another function of the system would be to provide the TPA user with special-case runs of RPA. These functions are included in the Three-Level Hierarchical System described next, and the functions will be elaborated upon in Section 6.

We will need to design a TPA-RPA interface module to manage the control of these functions and to ensure consistency. Data base consistency should not be a major problem, other than ensuring that the RPA target data base matches the TPA list of airfields.

This candidate architecture passes some of the criteria described in Section 5.1 but is not complete for non-F-111 planners. Thus, in our next architecture, we augment this architecture with KNOBS.

5.3.2.3 Three-Level Hierarchical System (TPA, KNOBS, RPA)

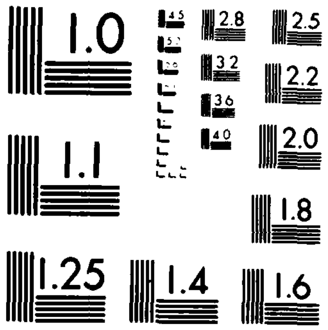
If we add KNOBS to the above system, we get a system which spans three levels of the planning hierarchy (target nomination, resource allocation (aircraft and ordnance), and route selection). KNOBS enables the user to use the integrated tool to perform weaponeering and aircraft assignment as well as target nomination, targeteering, and path selection.

The users of this three-level system will most likely be the Target Nomination Officer in the COID and the planner in Combat Plans. The Target Nominations Officer will have more information available to him than if he only had access to TPA. Thus he can make better decisions as to which targets to attack, given information on aircraft assignments and path selection. Similarly, the planner can make better decisions if he has access to RPA. The WOC planner would probably only have use for the RPA-module of the integrated aid, unless he wants to suggest alternatives to attacking assigned targets that he determines are well-protected by air defenses.

Many useful functions will be available to the user of this three-level aid. Constraint-checking, monitoring measures of effectiveness of the aids, better evaluation of cost, and the extension to non-F-111's

AD-A150 572   OCA (OFFENSIVE COUNTER AIR) MISSION PLANNING(U)   2/2
ADVANCED INFORMATION AND DECISION SYSTEMS MOUNTAIN VIEW
CA G COURAND ET AL. SEP 84 AI/DS-TR-3050-1
UNCLASSIFIED RADC-TR-84-189 F30602-83-C-0193   F/G 15/7   NL

END
FILMED
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A

are just a few of them. These functions are discussed in more detail in Section 6. Note that three interfaces -- TPA-KNOBS, KNOBS-RPA, and RPA-TPA will be needed to be built to perform these functions. A user interface will also be needed. Again, we discuss these interfaces in detail in Section 6.

This three-level hierarchical system passes all of the criteria presented in Section 5.3.1. With the addition of KNOBS, the system can extend to non-F-111's if desired. The details of just how this will occur is discussed in Section 6. There will, of course, be some pragmatic issues that must be addressed, such as data base consistency, but these issues can be resolved without too much trouble.

5.3.2.4 Augmented Three-Level Hierarchical System

If we add DAGR to the system described above, we do not add another level to the hierarchy, but we do augment the TPA level with a data base tool. DAGR updates the AOB file which TPA must access. Thus, by including DAGR in the integrated aid, we allow the AOB data base to be updated within the entire integrated system.

To include DAGR, we would need to build a DAGR-TPA interface module. This interface would be similar to what we described in Section 5.3.2.1 when we discussed the DAGR-TPA module of the Intelligence Tool. The interface would control the updates to TPA's AOB after DAGR is run. Data base consistency would be maintained through a Consistency Manager.

Although this candidate system allows more information to be used, the addition of DAGR in a sense makes the tool less complete. DAGR updates the AOB, but the AOB needs more information than DAGR currently uses to be as up-to-date as possible. Thus, if we add DAGR to the integrated system, we still need another tool to input more data to the

AOB. Thus, this augmented system does not pass our criteria of completeness.

6. SYSTEM DESIGN: TPA, KNOBS, RPA

The three-level hierarchical system - TPA, KNOBS, RPA - discussed in the previous section is selected to be the basis for the OCAP. We saw that this system passed each of our criteria for a valuable OCA planning system, and thus, we recommend that the three aids - TPA, KNOBS, RPA - be integrated into the OCAP. An overview of the OCAP is shown in Figure 6-1. This figure shows the flow of data, interfaces, and various users. It will be referred to throughout this section.

6.1 INTRODUCTION TO DESIGN

This section discusses the design issues raised in Section 5.2 in terms of a practical design of the OCAP. We recommend that this design be implemented when integrating the three aids - TPA, KNOBS, RPA. Sections are structured in parallel to those in Section 5.2.

The design of the data bases across the three aids will be discussed first. This section is followed by a description of the functions that the OCAP will serve. Then we discuss the different modes of planning a user might choose and how he can use the OCAP to serve those modes. The interfaces needed for the OCAP are discussed next, followed by man-machine interactions and user models. Finally, we discuss how the OCAP might evolve and what we need to consider for the future.

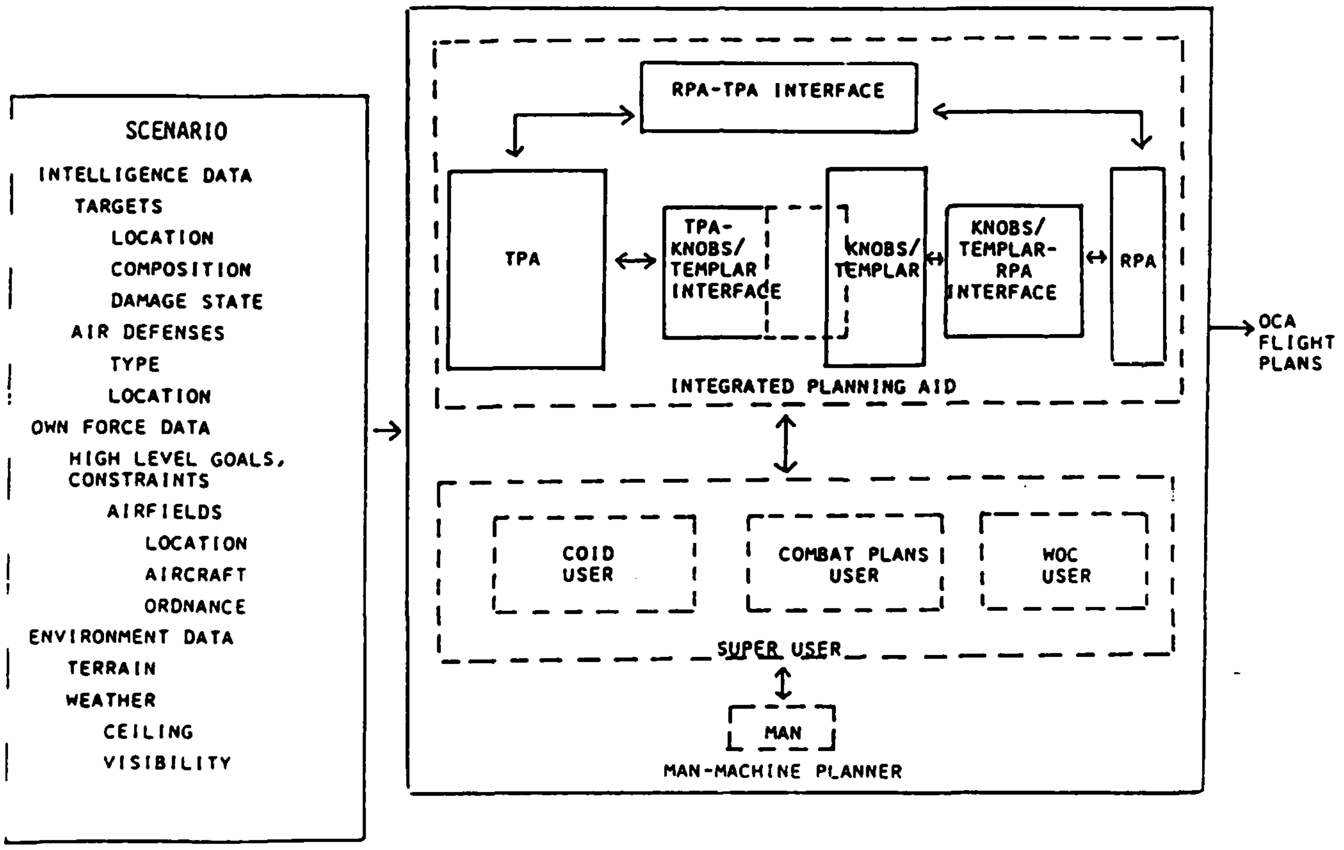


Figure 6-1: Hierarchical OCA Planning System

6.2 DATA BASES

Each of the three aids in the OCAP will access its own separate data base. Individual data bases per aid is necessary due to the software and hardware constraints of each aid. Thus, there will be redundant information stored across the aids' data base which must be kept consistent by a Consistency Manager as described in Section 4.

The scenario used for any run of the OCAP must be consistent across all data bases. Currently, each of the data bases include information for different scenarios. TPA contains 50 Warsaw Pact air bases. KNOBS targets are located primarily in East Germany. RPA contains test scenarios only, and thus they are probably not consistent with TPA or KNOBS. A common scenario must be selected for any set of runs of the OCAP, and each of the data bases must be updated accordingly.

The scenario data base will include three major divisions: intelligence data, own force data, and environmental data. These are shown in Figure 6-1. The intelligence data will include enemy targets (i.e., airfields), their locations, the components on each airfield, their damage states, and their locations. Intelligence data will also include air defenses' locations and types.

Own force data will include high level planning goals and constraints, airfields and their locations, numbers of each type aircraft, and amount of ordnance.

Environmental data will include terrain data that is required for RPA and the weather parameters of ceiling and visibility at the air bases.

Additionally, the scenario will have a script describing the state of the enemy airfields and the number of friendly aircraft available over a time line.

Note that because some of the data will be represented differently across data bases, the Consistency Manager must check consistency periodically while the OCAP is being used. It is not enough merely to ensure that the initial data bases are set up correctly at the start of the OCAP session.

6.3 FUNCTIONS OF THE OCAP

The primary function of the OCAP is to provide the pilot with the best set of targets and flight plans that are consistent with the available aircraft, ordnance and environmental conditions. The three aids - TPA, KNOBS, RPA - cover several of the important hierarchical planning functions required to achieve this goal. These functions are indicated in Figure 6-2.

The OCAP will allow input and planning expectations to flow smoothly from one level of the hierarchy to another. Figure 6-2 shows, in the left hand column, input data and constraints used by each of the planning functions. The arrows going down into each box indicate the planning assignments and expectations from the next higher-level functions. TPA receives its planning assignment from the user at the Intelligence level who has a set of goals in mind. KNOBS gets its planning assignment from TPA in the form of a prioritized list of targets and the amount of desired damage to the targets. Finally, RPA's planning assignment is a result of the aircraft-target assignments made by KNOBS.

The OCAP will also support feedback across the levels of the hierarchy. The arrows going up from each box in Figure 6-2 indicate feedback to a higher level. Feedback is in the form of the expected performance that can be achieved with the resources available and the applicable constraints. Feedback enables levels to perform their own functions better. For example, survivability feedback from RPA to TPA

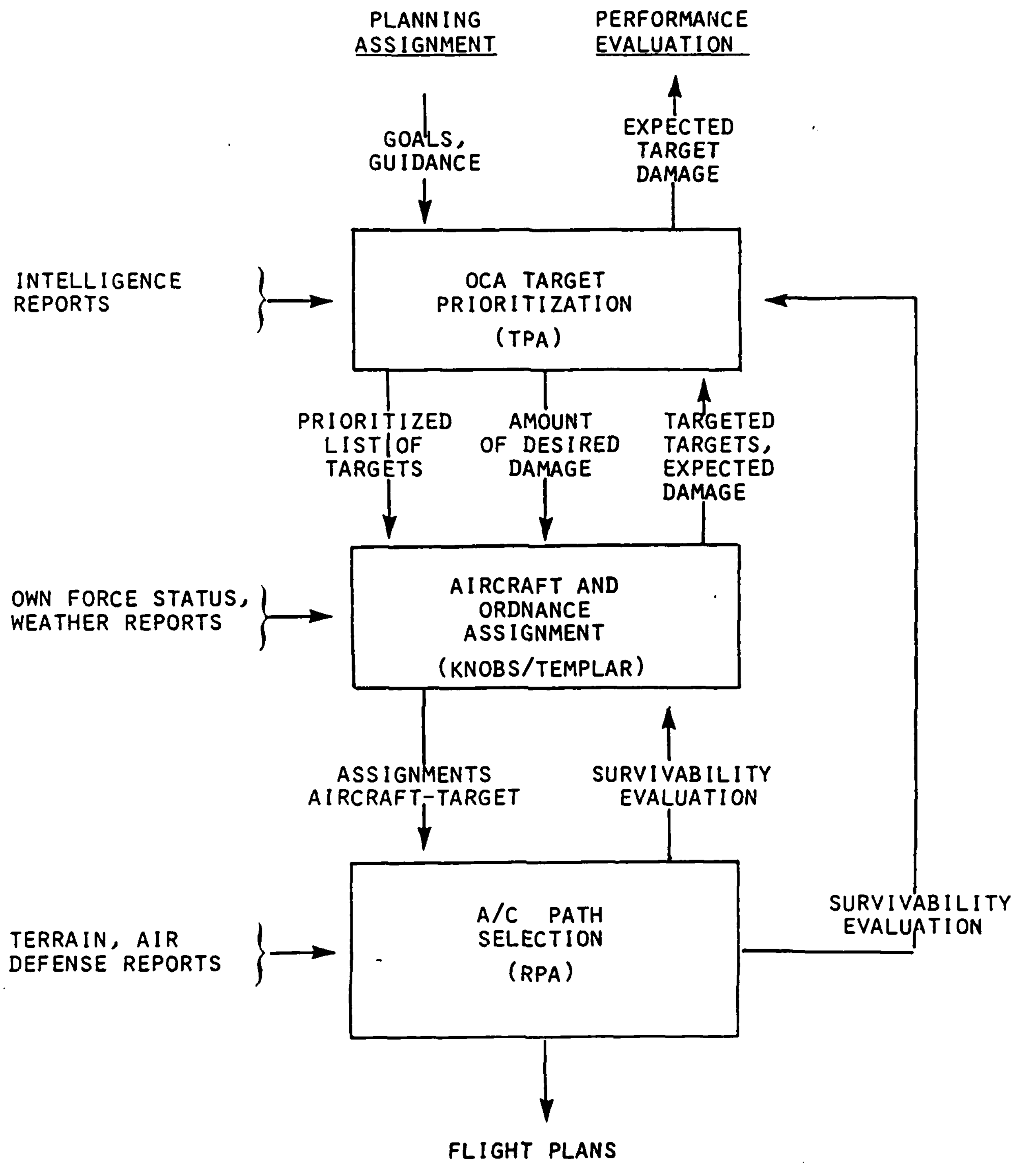

PLANNING ASSIGNMENT
PERFORMANCE EVALUATION
GOALS, GUIDANCE
EXPECTED TARGET DAMAGE
INTELLIGENCE REPORTS
OCA TARGET PRIORITIZATION (TPA)
PRIORITIZED LIST OF TARGETS
AMOUNT OF DESIRED DAMAGE
TARGETED TARGETS, EXPECTED DAMAGE
OWN FORCE STATUS, WEATHER REPORTS
AIRCRAFT AND ORDNANCE ASSIGNMENT (KNOBS/TEMPLAR)
ASSIGNMENTS AIRCRAFT-TARGET
SURVIVABILITY EVALUATION
TERRAIN, AIR DEFENSE REPORTS
A/C PATH SELECTION (RPA)
SURVIVABILITY EVALUATION
FLIGHT PLANS

Figure 6-2: OCAP Functions

enables TPA to adjust the cost it calculates of attacking a particular airfield. Feedback from KNOBS to TPA allows the user to compare overall measures of effectiveness computed by TPA with those actually achieved by KNOBS.

The computation and comparison of measures of effectiveness is a primary function of the OCAP. Measures of effectiveness are used in hierarchical systems for two purposes. First, they summarize and give data for the evaluation of a plan as it is shaping up. The second purpose is to facilitate a change in the flow of control in the hierarchical process. (The latter is essential to a truly hierarchical system.) For example, if the number of targets whose missions have low probability of survival values computed from RPA is large, the user may want to modify the TPA data base and then re-enter the planning system at the TPA level.

TPA computes the quantitative measure of effectiveness as the overall number of enemy sorties reduced. For good planning, the OCAP should provide the capability to record this and similar measures, such as percent reduction, for the geographic region which the enemy aircraft can reach and for the type of aircraft. These measures will be indicated over time. Qualitative measures of effectiveness for TPA will include some or all of the following:

1. The degree of consonance with apportionment and allocation decisions.

2. Overall maximum value versus value distribution over regions and time.

3. The sensitivity of a given quantitative measure of effectiveness as a function of the number of friendly aircraft available.

4. Types of enemy activities degraded (e.g., air-to-air, air-to-

ground, air-assault).

5. The number of high-value targets of opportunity that are covered (e.g., an enemy general is expected at airfield A tomorrow, so we could list it as a high-value target and get a report).

KNOBS/TEMPLAR measures of effectiveness will include some or all of the following:

1. Degree of consonance with overall apportionment and allocation numbers.

2. The number of constraints, according to priorities, not satisfied.

3. The number of high-value targets of opportunity covered.

RPA measures of effectiveness will include some or all of the following:

1. Probability of survival for individual missions.

2. Overall probability of survival for all missions planned.

3. Number of targets with reasonable value that have a mission probability of survival nearly equal to one.

The OCAP will compute and compare some or all of the measures listed below:

1. The TPA measures of effectiveness.

2. The TPA value-to-cost ratio, where the cost is related to the resources used and the probabilities of survival associated with the RPA measures.

3. The percent of individual missions that have a probability of survival exceeding an acceptable threshold.

4. The probability of survival averaged over all missions.

OCAP will support its data flow and feedback functions including calculation of measures of effectiveness, through interfaces between the aids. Section 6.5 will discuss the interfaces and their specific functions in much greater detail. These interfaces may be driven by different perspectives. These are discussed next.

6.4 PLANNING MODES

There are three basic perspectives from which one can plan: top-down/value-driven, resource-driven, and cost-driven. Each of these planning modes imply certain necessary characteristics of the interfaces: for example, whether the interface uses value or cost notions. The modes also imply different sequences of processing the planner would use while running the OCAP. These modes and processing sequences are described in this subsection. Following the description of the three modes is a discussion of composite perspectives of planning. This discussion combines all three planning modes to a more general approach.

6.4.1 Top-Down/Value-Driven Planning

Figure 6-3 indicates the flow of hierarchical planning activities when maximizing the value derived is the driving goal. The solid lines indicate positive flow and, in some sense, control of authority. The dotted lines represent feedback information of an alerting nature rather than directives. For example, if RPA indicates an extremely dangerous mission to a specific target, an alert would be sounded to KNOBS where the aircraft mission assignment was made. This alert is informative in the sense that it points out the degree of danger and possible loss of aircraft rather than stating that the mission cannot be flown. The double lines indicate data flow.

The sequence of processing for the top-down/value mode is as follows: TPA is run first, followed by KNOBS. If KNOBS finds a way of potentially getting better value by assigning aircraft to targets, not necessarily in the order of the original prioritization, it would feed this information back to TPA. (This type of feedback requires a capability, either in KNOBS or the TPA/KNOBS interface, to accept a ranking of targets based on the planner´s judgment, i.e., an ad-hoc value.)

For example, consider the data collected in Figure 6-4. For convenience, we have assumed that "value," as both TPA and KNOBS understand it, consists of numbers of points. Suppose that there are three targets, A, B, and C, and that TPA has assigned points (or the interface has) for each. Now suppose the planner has made judgments of value as shown. TPA´s values are the result of sortie reduction. The planner´s values arise from the following considerations: Target A is valued at 10 points. Target B is strategic - or perhaps Target C is just ´too far away.´ So Target B gets 15 points, even though little sortie reduction results from hitting it, and Target C just gets a few points. The point is that, assuming we only have enough resources to hit two targets, the goal of maximizing overall value leads to different target sets being chosen (by the planner as opposed to TPA). The feedback has thus informed TPA and/or the planner that alternative target rankings exist.

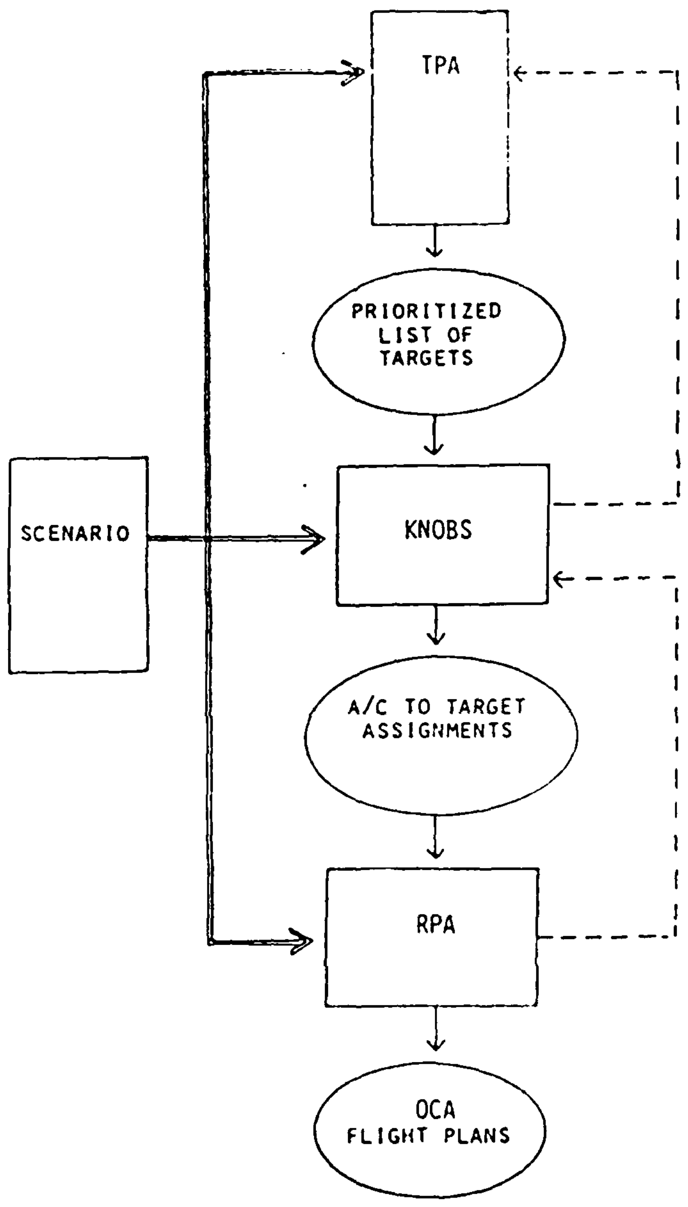


Figure 6-3: Top-Down/Value-Driven Planning

It will seek approval for one list, or even accept an entirely new list. The interface between KNOBS and TPA could start keeping statistics on which targets are being suggested, so that at the end of a planning process, the difference in the cumulative value derived from all targets targeted could be compared with the original values of targets provided by TPA.

KNOBS may also request TPA to be re-run again if, after running KNOBS, all the targets have been hit, but some aircraft are still available. This result would occur if the user had told TPA there were fewer aircraft than there actually were. Thus, TPA would need to be re-run with more aircraft, and the processing would continue.

After KNOBS is run, the aircraft-target assignments are passed to RPA. RPA computes the probability of survival to each target. If the survival is too low for some targets, and thus the expected value that can be achieved from those targets is much lower than TPA and KNOBS originally computed, RPA feeds this information back to KNOBS. KNOBS tries other aircraft assignments or returns flow to TPA, and the process is repeated.

6.4.2 Resource-Driven Planning

In resource-driven planning, (represented in Figure 6-5), the planner at the KNOBS level decides which targets to hit. He will use the nominated list of targets received from TPA as a basic guideline, but using the availability of critical resources, timing issues, and his own judgment, he (using KNOBS/TEMPLAR) will make the final decision as to which targets to hit. The TPA-KNOBS interface will then inform TPA of the revised prioritized list of targets. Thus, although TPA is run prior to KNOBS, KNOBS uses TPA output only as a basic guideline and not as a driving force.

TARGET / VALUE INITIATOR	A	B	C
TPA	10 PTS	3 PTS	12 PTS
THE PLANNER	10 PTS	15 PTS	5 PTS

- HAVE RESOURCES FOR 2 TARGETS
- TPA CHOOSES $\{A, C\}$. THE PLANNER'S VALUES LEAD TO THE TARGET-SET $\{A, B\}$.

Figure 6-4: TPA/KNOBS Value Trade-offs

The planner will consider all types of resources, including those which suppress air defenses, such as F-4G's, EF-111's, and tankers. The KNOBS-RPA interface can request RPA to determine the probability of survival to a specific target. From this value, the planner can judge whether air defenses must be suppressed before attacking the target. KNOBS can then determine which resources to use against the air defenses, and the interface will inform TPA that the air defenses will be hit. This interaction between KNOBS and RPA implies that KNOBS and RPA runs will be interleaved. That is, KNOBS will be interrupted occasionally for special-case RPA runs.

The timing issue considered in resource-driven planning is an important one. The planner does not want the activity at the air base to get too high. The flow of the aircraft in and out of the air base must be kept at a reasonable level. Thus, the planner will also use KNOBS/TEMPLAR to keep track of this flow and suggest alternative targets to hit if the target it was considering attacking must be hit with an aircraft from an air base which is overcrowded at the time. The timing issue can be supported by the flow of processing described above.

6.4.3 Cost Driven Planning

When cost or safety is the driving force, the feedback from RPA to KNOBS and TPA will take the form of a directive to change targets that have been nominated and/or prioritized. We call this cost-driven planning. It is represented in Figure 6-6.

The flow of processing which supports cost-driven planning is as follows. TPA is run first, followed by KNOBS/TEMPLAR. KNOBS determines the aircraft assignment based on the prioritized list of targets it receives from TPA. KNOBS then passes this list of aircraft-to-target assignments to RPA. RPA will then compute the best probability of

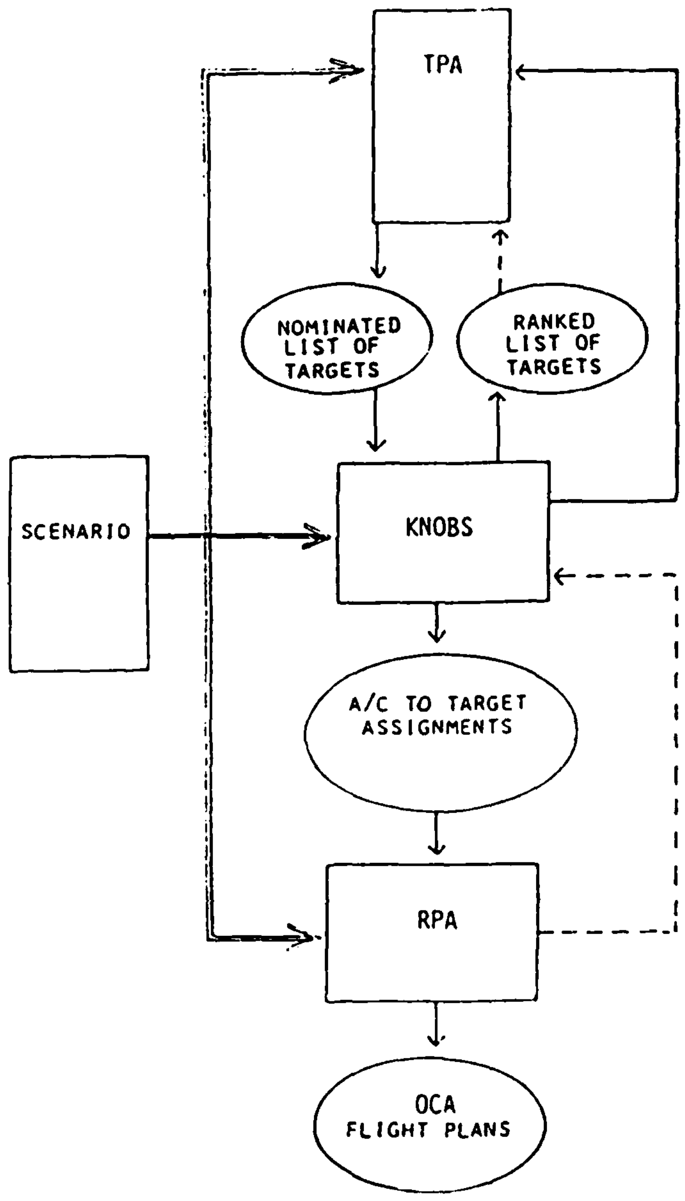


Figure 6-5: Resource-Driven Planning

survival to each target. If it is below some threshold, it will inform KNOBS that it needs a new assignment. Feedback will also flow up to the TPA level and suggest that the target list be revised. This process is repeated until RPA is satisfied with the list of targets, based on cost (safety) factors.

The planners at the WOC (RPA level) can also, upon determining that the route to a target is too dangerous, suggest other targets to hit. For example, they can suggest that air defenses be suppressed before attempting to attack a specific target. They can also use a coarse model of TPA and/or KNOBS to determine a revised list of targets to attack, compute the probability of survival to those targets, and feed suggested alternatives back to TPA and KNOBS.

6.4.4 Composite Perspectives of OCA Planning

In general, it is expected that the hierarchical planning process will respond to varying mixtures of the above three approaches to planning. The relative weight of these three approaches will probably, in fact, change during the planning process for any one day. Hence the architecture must have the flexibility to support the user as he changes perspectives and continues his planning process. This flexibility will be attained by designing the interfaces and the user environment support systems to accommodate all three of these major perspectives. Additional perspectives that are found to be important during the development of the aids, using the development scenario, will also be supported.

The flexibility of the architecture, in accommodating the three different perspectives, provides a capability to do the overall planning process in various ways. At one extreme it would accommodate a batch process. That is, the TPA activity is completed. A summary list of

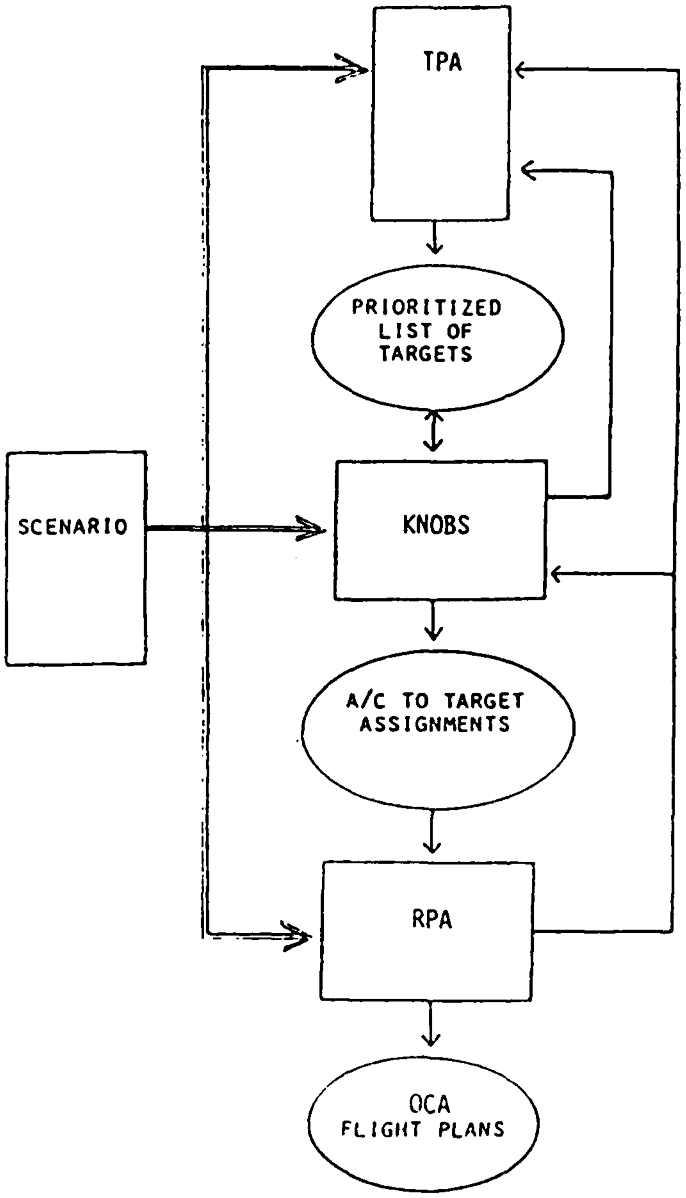


Figure 6-6: Cost-Driven Planning

targets is created for KNOBS to process one at a time. KNOBS completes its processing and then outputs a list of aircraft and target assignments for RPA to use in determining the individual flight plans. This type of (batch) process does not accommodate improving the overall OCA mission plan unless there is sufficient time to re-run each of the processes after anomalies have been discovered which indicate an important change. Unfortunately, current planning tends to resemble batch planning. This may be the result of the fact that different organizations are involved in the different parts of the plan. In part, it is also due to the lack of time to meaningfully consider many alternatives. However, the proposed hierarchical planning system does not have the same limitations as batch processing; this is the strength of the hierarchical planning concept.

We will now consider architectures that support the feedback and adjustment loops. Note that if these feedback and adjustment are "turned off," we have the batch process. The simplest architecture that uses feedback is represented by a case in which a new object, such as a new target or new resources, arises. This architecture is able to recognize such a condition, stop, do detailed planning if the planner wishes (on the new object), and then continue with what it was doing. This type of planning provides the capability to do lower level analysis before binding a decision at a higher level. For example, if an enemy airfield was being considered that had never been attacked before, the planner might wish to run an RPA process from the friendly airfield to the enemy airfield to get an assessment of the cost (the danger to our own fighter bombers). The TPA planner would not necessarily make a complete interactive run of the RPA system; he would only get the results of a single dynamic programming estimate of the probability of survival on the new target.

A more advanced architecture that uses feedback would not have to stop what it was doing should detailed planning be needed. It would be a decentralized architecture, perhaps based on a "blackboard" system (aids become knowledge sources, interfaces become blackboard managers and critics, etc.) The decentralized architecture is able to task a lower-level process to do a more complete analysis and report the result when it has it. Meanwhile, the higher-level planner continues with its own target considerations.

Note that this interaction between different levels in the planning hierarchy permits incremental adjustments to the overall planning process. This is contrasted with the batch procedure; if all adjustments are held at bay until the plan is completed, major modifications may be required.

6.5 INTERFACES

The key to integrating the basic functions performed in TPA, KNOBS and RPA into the OCAP lies in the interfaces. By making these interfaces "intelligent," we can produce summary information, set alerts, set controls for finding information on special case situations, and interact with the user at several levels in the planning process. These capabilities will allow a user to access the computer capability at any one of the functional levels in the hierarchy and obtain results at his level of expertise.

Figure 6-1 indicates the interfaces between TPA and KNOBS, KNOBS and RPA, and RPA and TPA. Through these interfaces, the user will be able to interact with each of the aids. Referring back to Figure 6-2, there is also a need for an interface to personnel and to organizations responsible for apportionment allocation and other high-level goal guidance and performance monitoring. The TPA/KNOBS interface will incorporate the functions that are required to interact with the higher-level planning personnel. The interaction with these higher-level planners within the hierarchical OCA planning system will take place through the user in the current model.

The flexible architecture discussed earlier will allow accessing any of these interfaces at any time the user wishes. This will permit the user to use the OCAP in the number of ways discussed in Section 6.4. In fact, it will allow the user, to some extent, to change his planning perspective back and forth.

In the future, the OCAP could include not only TPA, KNOBS, RPA or its equivalent, but also coarser models of each of these functions. Thus, a user would not necessarily have to interact closely with the functional model to get insight into a specific problem. For example, a coarse model for the RPA function might be an expert system that merely indicates the number of air defense systems an aircraft would fly over (or near) on a user-supplied route. It might have some thresholding scheme that would predict levels of danger depending on the numbers of air defense systems overflown. Thus, a planner at the TPA level could get a coarse estimate of probability of survival, without actually having to interact with RPA directly.

The following subsections indicate some of the capabilities that should be built into each of the interfaces of the OCAP.

6.5.1 TPA-KNOBS Interface

The TPA-KNOBS interface will be the most comprehensive. It will contain mechanisms for constraint-checking, for monitoring and assessing measures of effectiveness, and for translating TPA output to KNOBS input. These functions are described in this section.

6.5.1.1 Constraint Checking

The TPA-KNOBS interface will contain two types of constraint checking mechanisms. The first will exist outside of the core aids, while the second will use the current constraint checking mechanism already existing in KNOBS/TEMPLAR.

The first constraint checking mechanism will be used by external personnel and/or the user of the OCAP. While the user is running the system he, or another planner, may want to know the status of the plan. That is, he may ask how many resources have been assigned, from which bases, etc. Thus, he will want some summary information. The TPA-KNOBS interface will contain a constraint checking mechanism that keeps track of own forces and checks that constraints are not violated. This mechanism will be able to produce summary information for users to review and analyze.

The second type of constraint-checking mechanism exists within KNOBS currently. The interface will make use of this mechanism to keep track of own force data. Thus, if KNOBS attacks, say, only eight of the ten targets recommended by TPA, the interface will be able to feed back to TPA a list of targets that were not hit and the location of available resources. Note that, in general, we would like to keep the interfaces separate from the core aids. But since KNOBS and, later, TEMPLAR already contain constraint checking mechanisms, we can make use of them in this case.

6.5.1.2 Overall MOE Assessment

The interface modules will have mechanisms for monitoring and extracting parameters that accumulate into the measures of effectiveness discussed in Section 6.3. This monitoring capacity will occur for the current plan being developed, but it will also retain past days' plans for comparison trends and guidance for today's plan. Specific measures of effectiveness that the TPA-KNOBS interface will monitor include a set of parameters that accumulate into indications of the degree of consonance that is being attained with the apportionment and allocation decisions. This will be done for the overall case, i.e., all geographic regions as well as by each geographic region of interest. (There are two parts to this geographic compartmentalization, (1) enemy areas and (2) the friendly army areas which are covered by various enemy aircraft from enemy air bases within range.)

An example of how the TPA-KNOBS interface will assess measures of effectiveness follows. A specific MOE computed by TPA is total enemy sorties reduced. The TPA-KNOBS interface will accumulate this measure as each mission is scheduled. This cumulative effectiveness will be compared to the original measure computed by TPA. If the cumulative value is less than the original, the interface will alert the user. The user should then decide if he wants to replan the missions.

Additionally, the interface will have received a list of special targets from a user at the beginning of the planning session. These targets will be checked off as they are planned by KNOBS. The user will be notified if these targets have not been planned by the time the number of missions within the region containing the specific target reaches or exceeds the number of missions allocated for that region.

6.5.1.3 Conversion of TPA Output to KNOBS Input

TPA and KNOBS process information from two different perspectives. Given the number of F-111 sorties for OCA, TPA will find the best set of target components to attack, i.e., given cost, find the best benefit. KNOBS, however, performs its functions conversely. Given a specific benefit (i.e., Pk for the target component under attack), KNOBS finds the best cost solution, i.e., the number of aircraft required to produce the damage desired. Thus, for each target component on TPA´s target prioritization list, KNOBS needs a Pk to use as input. KNOBS also requires a time-over-target (TOT), which TPA does not use or compute. Thus, a translation must be performed to form TPA output to KNOBS input. (This will be part of the TEMPLAR processing task rather than an input).

The TPA-KNOBS interface module will produce a Pk for the target component in one of the following ways:

1. Table Lookup. There are ten components to the prototypical air base, each component can be in any one of five damage states. The TPA will have identified the component, the existing damage state, and the number of F-111 aircraft attacking this component. There are fifty component/damage state possibilities. If we allow up to ten F-111s on any one mission, then there are a maximum of 500 Pk´s that would be possible. This first option for calculating Pk would require JMEMS (Joint Munitions Effectiveness Manual System) to produce a table of Pk parameters for all 500 entries off-line. The KNOBS process would continue for the remaining aircraft to be assigned and the new target list that TPA provides.

2. Real-time Computation. The second option allows the user, while running the planning system, to leave the OCAP and run JMEMS to compute the Pk for only the specified target in which he is interested. This option has the advantage of computing only the Pk´s that are necessary. But it requires user interaction.

3. User Provides Pk. Finally, the OCAP could prompt the user to provide a Pk. The user can use his discretion in determining the value.

The TPA-KNOBS interface will compute a time-over-target, to use as input to KNOBS, in two different ways, depending on aircraft type.

1. F-111's. An F-111 has a sortie generation rate of one per day. Thus, there are at least two ways the interface could determine a TOT. First, it could simply select a TOT, over the 24-hour period, at random. Choosing a random time makes sense from the planner's point of view, because he does not want the enemy to guess his sortie schedule. A second method for computing TOT is by constraint checking. The interface would keep track of the flow of aircraft out of each base and determine a TOT that is consistent with this flow. This method would ensure that air base runways do not get overcrowded at any one time.

2. Non-F-111's. Aircraft other than F-111's, such as A-7's, have sortie generation rates of more than one per day. (Note that a flight day for A-7's consists of about 12 hours during the day. They cannot fly at night, in contrast to the F-111's.) Thus, the interface will compute a "turn time"; that is, the total hours of flight day divided by the sortie generation rate. Then the interface will determine a TOT over the turn time in the same way as for F-111's.

6.5.1.4 Summary of TPA-KNOBS Interface Module Characteristics

The characteristics of this interface are enumerated in Figure 6-7.

These characteristics permit the user to exploit some of the hierarchical characteristics of the major planning functions required to plan effective OCA missions.

1. CONSTRAINT CHECKING
   - PROVIDES SUMMARIES TO USERS
   - PROVIDES FEEDBACK FROM KNOBS TO TPA

2. MOE ASSESSMENT
   - MONITORS PERFORMANCE
   - ACCUMULATES MEASURES OF EFFECTIVENESS
   - ALERTS USER IF CUMULATIVE MOE IS TOO LOW

3. TPA OUTPUT $\longrightarrow$ KNOBS INPUT
   - PK
     - TABLE LOOKUP
     - REAL-TIME COMPUTATION
     - USER PROVIDES PK
   - TOT
     - RANDOM
     - CONSTRAINT CHECKING

Figure 6-7: Summary of TPA-KNOBS Interface Characteristics

6.5.2 KNOBS-RPA Interface

The KNOBS-RPA interface will perform at least four basic functions: (1) control special-case RPA runs requested by KNOBS, (2) alert the user when particular mission probabilities of survival are too low, (3) build a summary of probability of survival, and (4) develop the OCA lines of the ATO. This section describes each of these functions.

Note that the current RPA gives us the probability of survival for an F-111 flying from the friendly air base to the designated target. This probability of survival could be used as an approximate probability of survival for any of the other aircraft types. However, since the F-111 has terrain following capabilities, the KNOBS-RPA interface module would want to somehow disable this function in RPA, possibly by constraining the aircraft to fly at a fixed altitude. This would give a more realistic probability of survival for the other types of aircraft that did not have terrain following. (e.g., F-4, F-16, A-7). This amount of accuracy would probably suffice for the planners working at higher levels in the hierarchy, other than the WOC.

6.5.2.1 Special-Case RPA Runs

The KNOBS-RPA interface will control the processing of special-case RPA runs. It will receive requests from KNOBS when special case considerations are being analyzed in the KNOBS process. That is, if the KNOBS process wants a refined estimate of the probability of survival to a particular target from a particular friendly air base, the KNOBS-RPA interface module will be notified. The interface will then set up and control an RPA run to calculate the desired probability of survival. This RPA calculation could be controlled by the interface module to provide an automatic answer by simply using the dynamic programming subset of RPA. Or it could compute a more refined estimate by requesting the user to interact with the RPA model and then to tell the interface the resulting probability of survival.

6.5.2.2 Alerting the User

The interface module would also receive alerts from the RPA planning process when aircraft-to-target assignments result in a mission whose probability of survival is beyond a reasonable threshold according to the Wing's guidelines for safety. This alert would be used to alert the user about the low probability of survival. The interface would keep track of the resources and ordnance left to be scheduled so that the user can re-run KNOBS for that particular target if he so desires.

6.5.2.3 Summary of Probability of Survival

This interface will also build a summary of the friendly aircraft survival expected over all of the missions. This summary will probably be available for the user to examine at any time.

6.5.2.4 Build ATO

The KNOBS-RPA interface module will incrementally develop the OCA lines of the ATO that will be transmitted to the Wing Operation Centers via CAFMS (Computer Aided Force Management System). The collection of these ATO line items builds a summary, as the hierarchical planning process progresses. The summary information can be used for review by the user. In the future, the CAFMS software for sorting on various parameters, could also be inserted in this module to help the review process.

6.5.3 RPA-TPA Interface

The RPA-TPA interface will provide two major functions. It will furnish TPA with expected survival capabilities which TPA can use to adjust its cost. Also, the interface will process and control special-case runs of RPA. These functions are described in this section.

6.5.3.1 Adjusting TPA's Cost

The RPA-TPA interface will provide TPA with expected survival capabilities to each of the enemy airfields. (For most planning purposes at the TPA level, a rough approximation to survival is sufficient.) This estimate of survival could be pre-calculated using the dynamic programming part of RPA to calculate the probability of survival from each friendly airfield to each enemy airfield of interest. This pre-calculation could be done once at the beginning of the planning cycle, after the new enemy air defense Order of Battle information is available. The results of these pre-calculations would be stored in a table for lookup upon request by the TPA process. (Note that this pre-calculation may not have to be done completely each day if the enemy air defense situation changes only slightly.)

The planner, through one of the user interfaces, could use these pre-calculated survivability calculations to adjust the number of aircraft suggested by TPA for a particular air component. Or, the RPA-TPA interface could adjust the number automatically. For example, if TPA determined that four aircraft should be sent to an enemy air's POL site, and the RPA-TPA interface indicated the probability of friendly aircraft surviving the flight to the airfield were .8, then the total number of aircraft that should be assigned to achieve four at the target would be 4/.8=5 aircraft. This use of low-level information in the higher TPA level of the planning hierarchy provides a capability primarily for Intelligence personnel. They are able to make more accurate estimates of the effects of survivability which the pilots are likely to use.

6.5.3.2 Special-Case Runs

The other major function of the RPA-TPA interface module will be to provide data to the TPA process in response to special requests. The planner using TPA could, for example, put in a special request for a refined estimate of survivability to the enemy airfield if he feels the air defenses have changed significantly or if he is targeting an enemy air base that has not been targeted before. In these special cases, the interface coulo either (1) run the dynamic programming portion of RPA only or (2) interact through the user interface to involve a planner familiar with the RPA process to give a detailed result for the expected probability of survival.

This function is similar to the function the KNOBS-RPA interface will perform, described in Section 6.5.2.1.

6.5.4 Scenario Data Base - Core Aids

Although Figure 6-1 does not indicate explicitly an interface between the scenario data and the various aids and interface modules, it is clear that such an interface must exist. However, this inferface is very simple and its function will only be to convert the scenario data into the specific formats that each of the aids and interface modules require. This must be done at the initialization time for running the entire planning process and at any later time when the scenario data is changed.

6.6 USER MODELS

The hierarchical OCA planning architecture allows the human planner to interact with the computer system through a set of user interfaces. As indicated in Figure 6-1, there are four user core models which these interfaces support. The four models are (1) the COID user, (2) the Combat Plans user, (3) the WOC user, and (4) the super-user. (Super-user means the human planner can play a composite of the other three roles in doing his planning.)

The user models provide a way for the user to interface with the system at different levels to achieve different functions. A user model is a domain-specific technique for tailoring the planning system´s functions to a given user. In other words, the planning system is composed of many functions, and user models will define the access, processing, and output that a given user type sees for a given function.

The user of summary information is an important feature of the user models. The interface modules will contain the computer support necessary for the user models to provide summary information. These summaries do two things. They abstract essential elements of cases that have been run and use the abstraction to set up other processes. It is possible for the abstracted case to then run automatically. For example, the summary might cause an input file to be modified and an aid that is higher in the hierarchy to be run. The summaries are also responsible for accumulating statistics; these statistics are used to evaluate the OCA portion of the ATO. Hence, the summaries have an active feature - creating cases to be investigated, and a passive feature - collecting data for off-line evaluation.

Summaries will be the primary medium through which the human interacts with the aid. In other words, they will be the primary interface to the user (or user environment) when he is initiating a TPA, KNOBS, or RPA process. Different user models will provide different levels of summary information to the planner.

The four user models are described next, followed by a description of the man-machine interface. This interface is required to enable the user to interact with the OCAP while using a particular user model.

6.6.1 COID User Model

This model will provide a set of menus and interaction protocols for the user who is primarily interested in the target nomination and targeteering levels within the hierarchical planning activities. This user will be looking at the planning process primarily from the perspective of an Intelligence Officer interested in determining a prioritized target nomination list, enemy air defense capabilities, and time-line effects on enemy capability. He will also be interested in weaponeering to the extent of estimating the amount of ordnance required to achieve a desired level of damage to the targets on the prioritized target nomination list. Thus, the COID user model will provide for ease in interacting with TPA, the TPA-KNOBS interface module, and the RPA-TPA interface module. In the future, this user model could become an expert system and anticipate the types of interaction in which the COID user would or should be involved.

The COID user model will allow the user to access all levels of the OCAP. Detailed summaries will be available to him at all levels.

6.6.2 Combat Plans User Model

The Combat Plans user model will be effectively the same as the model for the COID user. The main difference is that this model will provide ease of interaction between the human user and KNOBS, the TPA-KNOBS interface module, and the KNOBS-RPA interface module. This user model will also be more concerned with detailed summary information about KNOBS and the above-mentioned interfaces. It would allow the planner to change data items concerned only with the commitment of resources.

6.6.3 WOC User Model

This user model will provide ease of interaction between the human and the RPA, the KNOBS-RPA interface module, and the RPA-TPA interface module.

The WOC user will have access to all features of the RPA-module, but not all features of the TPA and KNOBS-module. For example, the WOC user could not change the status of a target, but he could suggest that specific targets be attacked. The WOC user model will provide the planner with detailed summaries of RPA MOE's, but would not bother with such information concerning TPA or KNOBS.

6.6.4 Super-User Model

This model will provide equal access to all of the core models and interface models. In the future, expert systems knowledge could be provided that concentrates on tradeoffs among the levels of the hierarchy and allows the user to explore the effects of summary information and special case planning items that arise.

6.6.5 Man-Machine Interface

For the user's point of view, it will not make any difference what computers are being used for the individual aids and interface modules. The user should only be concerned about performing the functions he wants and needs in order to accomplish the hierarchical planning process. Thus, the man-machine interface must take care of all of the systems issues of interacting with the appropriate machinery and controlling it appropriately in a relatively foolproof manner. That is, inadvertent or naive entries by the human should not cause the program to quit functioning or to get into a line of reasoning that is not profitable. The "user friendliness" will be provided primarily through the user models described above.

The primary interface devices will consist of a color graphics capability, a regular terminal, and a keyboard with special functions buttons programmed. In the future, models may include voice interaction and/or touch screen displays.

The color graphics capability will be similar to that which will be developed during the TEMPLAR project. In particular, it will display geometric data indicating location of air bases and the specific components that are being considered for attack at each enemy airfield. These components will be color coded. The regular computer terminal will be used for menus and to provide input and output of symbolic and numerical data. Function keys will be provided for input and output requests that occur frequently. The natural language capability in KNOBS need not be extended or used extensively if at all.

Because the main purpose of this research and development effort is to explore ways to do good hierarchical planning, the man-machine interface for the integrated hierarchical OCA planning system will emphasize the fundamental concepts of how to build a good plan. The three core aids of TPA, KNOBS and RPA have each developed a research-style man-machine interface. The man-machine interface for the integrated hierarchical OCA planning system will continue in this research vein. The man-machine interface will be developed only to the extent that it allows planners to build good plans with ease.

6.7 EVOLUTIONARY CONSIDERATIONS

A primary consideration in our practical design of the OCAP is evolution. If one or more of the three aids evolves or gets replaced by a tool which performs similar functions, what will happen to the OCAP? We have designed the system in such a way that the answer to this question is that only minor modifications need be made.

The design of each of the interfaces to be outside of the specific core aids was a direct result of this evolutionary consideration. The use of KNOBS constraint-checking mechanism is the only exception to all of the interfaces being external to the aids. This exception is not a problem because KNOBS successor, TEMPLAR, will also contain these constraint-checking mechanisms.

Because of our involvement in TEMPLAR, we have explicitly considered it in our design of the OCAP. This consideration will allow for ease in replacing KNOBS in the future.

Data base consistency is another issue that must be covered when considering evolution. Again, our design of the OCAP architecture explicitly allows for evolution of the data bases. The use of a Consistency Manager, as discussed in Section 4, does not constrain the aids or their data bases to be static.

In summary, the system design of the OCAP allows the core aids - TPA, KNOBS, RPA - to improve or even to be replaced by tools which perform similar functions. Minor modifications to the interfaces will be necessary, but the software of the aids need not be changed.

8. McCune, B.P., Shapiro, D.G., Wishner, R.P., Fehling, M.R., Payne, J.R., Kefalas, J., Aldrich, J.R., Albritton, J.P., and Adams, J.E., (1983). Tactical Expert Mission Planner (TEMPLAR). AI&DS Proposal No. 8502, Advanced Information & Decision Systems.

9. Millen, J.K., Engelman, C., Scarl, E.A., and Pazzani, M.J., (no date). KNOBS Architecture.. The MITRE Corporation.

10. Riemenschneider, R.A. and Rockmore, A.J., (1983). Route Planning Aid: User's Manual.. Systems Control Technology, Inc.

11. Riemenschneider, R.A., Rockmore, A.J., and Wikman, T.A., (1983). Route Planning Aid: System Specifications. Systems Control Technology, Inc. and PAR Technology Corporation.

12. Rockmore, A.J., Riemenschneider, R.A., and Wikman, T.A., (1983). Route Planning Aid: Functional Description.. Systems Control Technology, Inc. and Par Technology Corporation.

13. Waslov, K. A., (1982). Target Prioritization Aid: Users Guide.. Decisions and Designs, Inc.